Document Classification with Latent Dirichlet Allocation

István Bíró

Supervisor: András Lukács Ph.D.

Eötvös Loránd University
Faculty of Informatics
Department of Information Sciences

Informatics Ph.D. School
János Demetrovics D.Sc.

Foundations and Methods of Informatics Ph.D. Program
János Demetrovics D.Sc.

Budapest, 2009

Alulírott Bíró István kijelentem, hogy ezt a doktori értekezést magam készítettem és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint, vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Budapest, 2009. június 30.

Bíró István

Az értekezés bírálatai és a védésről készült jegyzőkönyv megtekinthető az Eötvös Loránd Tudományegyetem Informatikai Karának Dékáni Hivatalában.

Acknowledgements

Firstly, I wish to thank my supervisors András Lukács and András Benczúr for introducing me to the topic of data mining. Without their constant support and guidance this thesis would have never been completed.

Among my colleagues, I am especially grateful to Jácint Szabó for the successful research that we did together. These results have finally become the main constituents of my thesis. I also express my acknowledgements to Csaba Szepesvári and Zoltán Szamonek for the joint work in my language modeling research. I would like to thank all of my co-authors for the successful team work and the numerous fruitful discussions. I am grateful to the Data Mining and Web Search Group in MTA SZTAKI, which proved to be a pleasant and inspiring environment. I would like to thank Ana Maguitman for the invitation to the University of South at Bahia Blanca as well as for the Argentine helpfulness and support during the three months research period. I am much in debt to András Benczúr, Mátyás Brendel and Jácint Szabó for reading my thesis and providing useful comments and suggestions.

My research was partially supported by the projects EU FP7 JUMAS, NKFP-07-A2 and NKFP ASTOR 2/004/05 projects and the grant OTKA NK72845 of the Hungarian National Science Fund.

Last but not least, I would like to say thank you for the trust and love of my wife Ari and my two daughters Virág and Dorka providing me half of the success in achieving my goals.

Notation

$\mathcal{D}$	document collection (corpus), $\mathcal{D} = \{d_m\}_{m=1}^{M}$
M	number of documents in the corpus
$\mathcal{V}$	vocabulary, $\mathcal{V} = \{t_j\}_{j=1}^{V}$
V	number of terms in the vocabulary
$\mathcal{K}$	set of latent topics
K	number of latent topics
N	number of words (word-positions) in the corpus
N_m	document length, $N = \sum_{m=1}^{M} N_m$
$\vec{\alpha}, \vec{\beta}, \vec{\gamma}$	Dirichlet hyperparameters
$\vec{\vartheta}_m$	topic mixture vector for document d_m
$\underline{\Theta}$	topic mixture matrix for the whole corpus, $\underline{\Theta} = \{\vec{\vartheta}_m\}_{m=1}^{M}$
$\vec{\varphi}_k$	word mixture component of topic k
$\underline{\Phi}$	word mixture matrix, $\underline{\Phi} = \{\vec{\varphi}_k\}_{k=1}^{K}$
$z_{m,n}$	mixture indicator that chooses the topic for the n^{th} word in document d_m
$w_{m,n}$	term indicator for the n^{th} word in document d_m

Abbreviations

AUC	Area Under Curve (related to the Receiver Operator Characteristic)
IR	Information Retrieval
F	F-measure
LDA	Latent Dirichlet Allocation
LSA/LSI	Latent Semantic Analysis/Indexing
ODP	Open Directory Project
PLSA/PLSI	Probabilistic Latent Semantic Analysis/Indexing
SVD	Singular Value Decomposition
SVM	Support Vector Machine

Contents

Chapter 1

Introduction

1.1 Overview

One of the main consequences of the "Age of Internet" is the enormous amount of electronic data, mostly in the form of text. In the last two decades we observe increasing need for very efficient content-based document management tasks including document classification (or alternatively text categorization or classification), the process of labeling documents with categories from a predefined set. Although the history of text categorization dates back to the introduction of computers, it is only from the early 90's that text categorization has become an important part of the mainstream research of text mining, thanks to the increased application-oriented interest and to the rapid development of more powerful hardware. Categorization has successfully proved its strengths in various contexts, such as automatic document annotation (or indexing), document filtering (spam filtering in particular), automated metadata generation, word sense disambiguation, hierarchical categorization of Web pages and document organization, just to name a few.

The efficiency, scalability and quality of document classification algorithms heavily rely on the representation of documents. Among the set-theoretical, algebraic and probabilistic approaches, the vector space model [101] representing documents as vectors in a vector space is used most widely. Dimensionality reduction of the term vector space is an important concept that, in addition to increasing efficiency by a more compact document representation, is also capable of removing noise such as synonymy, polysemy or rare term use. Examples of dimensionality reduction include Latent Semantic Analysis (LSA) [31] and Probabilistic Latent Semantic Analysis (PLSA) [58]. See details in Chapter 2.

Another interesting approach that can be suitable for low dimensional representation is a generative probabilistic model of text corpora, namely **Latent Dirichlet Allocation (LDA)** by Blei, Ng, Jordan [13]. LDA models every topic as a distribution over the words of the vocabulary, and every document as a distribution over the topics, thereby one can use the latent topic

mixture of a document as a reduced representation. LDA has several applications including entity resolution [11], fraud detection in telecommunication systems [120], and image processing [38, 110], in addition to the large number of applications in the field of text retrieval. For more details about LDA see Chapter 3.

1.2 Our contribution

In the present thesis we discuss our results and experiments with LDA. Firstly, we describe a thorough comparison of LSA and LDA in Web content classification. Secondly, we modify the original LDA model in order to be applicable to supervised classification. Thirdly, we propose a novel influence model that gives a fully generative model of the document content taking linkage into account. Fourthly, we develop LDA specific boosting of Gibbs samplers resulting in a significant speedup in our experiments. Finally, we apply our models to the task of Web spam filtering at the Web Spam Challenge 2008. In the rest of this section we present our main contribution separately in the next five subsections.

1.2.1 Comparative Analysis of LSA and LDA

In Chapter 4 we discuss the application of two text modeling approaches, LSA and LDA for Web page classification. We report results on a comparison of these two approaches using different vocabularies consisting of links and text. Both models are evaluated using varying numbers of latent topics. Finally, we evaluate a hybrid latent variable model that combines the latent topics resulting from both LSA and LDA. This new approach turns out to be superior to the basic LSA and LDA models.

We study the classification efficiency of the LSA and LDA based latent topic models built on different vector space representations of web documents. Generally, the vector space is spanned over documents words. We refer to this representation as *text based*. However, in a hyperlinked environment, one can represent documents with their in- and outneighbors as well, naturally defining a *link based* representation.

Experimental results

We conduct a series of experiments to compare LDA to LSA, based on both text and link based representations of Web documents. We use a thoroughly prepared subset of the ODP Web directory. To a given page, we assign its main category label from the ODP category hierarchy. After generating our models, we use the Weka machine learning toolkit with 10-fold cross validation to run two different binary classifiers: C4.5 and SVM, separately for each category. The key results are as follows:

- LDA outperforms LSA by 8% in *F*-measure and 5% in *AUC*.
- Link based representation is weaker than text based, however the combination of the two performs the best. This reveals the possibility that different representations capture different pieces of information hidden in the documents. The improvement of a text and link based representation over a simple text based one is 27% in *F*-measure and 7% in *AUC*, respectively.
- LDA and LSA models capture different aspects of the hidden structure of the corpus and the combination of these models can be highly beneficial. The improvement of an LDA-LSA hybrid model over LSA is 20% in *F*-measure and 7% in *AUC*, while over LDA is 11% in *F*-measure and a 2% in *AUC*.

The results appeared in Proceedings of the 6^{th} IEEE Latin American Web Conference (*LA-WEB 2008*).

1.2.2 Multi-corpus LDA

In Chapter 5 we introduce and evaluate a novel probabilistic topic exploration technique, called **Multi-corpus Latent Dirichlet Allocation (MLDA)**, for supervised semantic categorization of Web pages. The appealing property of MLDA is that right after unseen inference the resulting topic distribution directly classifies the documents. This is in contrast to, say, plain LDA, where the topic distribution of the documents serves as features for a classifier.

MLDA model

MLDA is essentially a hierarchical method with two levels: categories and topics. Assume we have a supervised document categorization task with m categories. Every document is assigned to exactly one category, and this assignment is known only for the training corpus. For every category we assign separate topic collection, and the union of these collections forms the topic collection of LDA. In LDA a Dirichlet parameter vector α is chosen for every document so that topics assigned to the document words are drawn from a fixed multinomial distribution drawn from $\mathrm{Dir}(\alpha)$. In MLDA, we require for every training document that this Dirichlet parameter α has component zero for all topics outside the document's category. In this way, only topics from the document's category are sampled to generate the document's words. This is equivalent to building separate LDA models for every category with category-specific topics. For an unseen document d the fraction of topics in the topic distribution of d that belong to a given category measures then how well d fits into that category. As a Dirichlet distribution allows only positive parameters, we will extend the notion of Dirichlet distribution in a natural way by allowing zeros. Although there exist hierarchical latent topic models [14, 111] to tackle more than one

layers similar to ours, the advantage of MLDA is that it is built up from plain LDA and no complicated hierarchical models need to be developed.

Vocabulary term selection and ϑ-smoothing

We perform a careful term selection based on the tf-value of terms calculated for each different category and on the entropy of the normalized tf-vector over the categories. The goal is to find terms with high coverage and discriminability. There are several results published on term selection methods for text categorization tasks [42, 67]. However, we do not directly apply these here, as our setup is different in that the features put into the classifier come from discovered latent topics, and are not derived directly from terms. We introduce a heuristic, ϑ-smoothing which handles the effect that similar topics may arise in distinct categories.

Experimental results

We use the same Web document collection as in Section 1.2.1. We compare the classification performance of MLDA with that of several baseline methods. We also test the effectiveness of our proposed term selection and ϑ-smoothing. In the MLDA model, one can easily exploit the statistics of the hierarchical category structure of the Web collection, therefore experiments on this feature are done as well. MLDA can be used as a direct classifier, since the inferred topic distribution gives an estimation of how well the document fits into the category. The key results are as follows:

- Improvement in running time over LDA. In addition MLDA can be run in parallel.
- 15% increase in *AUC* with MLDA based direct classification over $tf \times idf$ based linear SVM.
- A slight 2% increase in *AUC* by choosing the number of subcategories of the main category as the number of latent topics.
- 10% increase in *AUC* by combining a careful vocabulary selection and the ϑ-smooting heuristic over a "heuristic-free" MLDA baseline.

The results are accepted to the 19^{th} European Conference on Machine Learning and 12^{th} Principles of Knowledge Discovery in Databases (*ECML/PKDD 2009*).

1.2.3 Linked LDA

In Chapter 6 we propose **linked LDA**, a novel influence model that gives a fully generative model for hyperlinked documents, such as a collection of Web pages. We demonstrate the

applicability of our model for classifying large Web document collections. In our setup, topics are propagated along links in such a way that linked documents directly influence the words in the linking document. The inferred LDA model can be applied to dimensionality reduction in classification similarly to results in Subsections 1.2.1 and 1.2.2. In addition, the model yields link weights that can be applied in algorithms to process the graph defined by Web hyperlinks; as an example, we deploy link weights extracted by linked LDA in stacked graphical learning [63].

Experimental results

In our experiments, we use the host-level aggregation of the WEBSPAM-UK2007 crawl of the `.uk` domain. We make the following experiments in order to study the behavior of the linked LDA model:

- We experimentally show the convergence of the log-likelihood of the model.
- We experimentally show a strong correlation between the model log-likelihood and the *AUC* of classification.
- We compare linked LDA with the classical LDA model: linked LDA outperforms LDA by about 4% in *AUC*.
- We compare linked LDA with link-PLSA-LDA, the state of the art link based topic model. In this experiment, we use different link weight functions derived from the χ distribution of linked LDA, from link-PLSA-LDA and the cocitation graph as well. We perform the experiments with a stacked graphical learning classification scheme. The best model is linked LDA; it improves the classification over link-PLSA-LDA by 3% in *AUC*.

The results are submitted to the ACM 18^{th} Conference on Information and Knowledge Management (*CIKM 2009*).

1.2.4 Fast Gibbs samplers

In Chapter 6 we develop LDA specific boosting of Gibbs samplers resulting in a significant speedup in our experiments. The crux in the scalability of LDA for large corpora lies in the understanding of the Gibbs sampler for inference. Originally, the unit of sampling or, in other terms, a transition step of the underlying Markov chain, is the redrawing of one sample for a single term occurrence in the Gibbs sampler to LDA [47] as well as in the fast Gibbs sampler [91]. The storage space and update time of all these counters prohibit sampling for very large corpora. Since however the order of sampling is indifferent, we may group occurrences of the same term in one document together. Our main idea is then to re-sample each of these term positions in one step and assign a joint storage for them. We introduced three strategies: for

aggregated sampling we store a sample for each position but update all of them in one step for a word; for *limit sampling* we update a topic distribution for each distinct word instead of drawing a sample, while for *sparse sampling* we skip some of these words at random.

Results

- We prove that the heuristics have rationale, because they all have the required distribution $p(\vec{z}|\vec{w})$ as their unique stationary distribution.
- We compare the running time of the proposed heuristics with the standard Gibbs Sampling and with a recently developed fast Gibbs Sampler [91]. We measure an astonishing $10 - 11$ times speedup over the fast sampler with only 1% decrease in terms of *AUC*.
- We experimentally show the convergence of the model log-likelihood of the various samplers and the strong correlation between the model log-likelihood and the *AUC*.

The results are submitted to the ACM 18^{th} Conference on Information and Knowledge Management (*CIKM 2009*).

1.2.5 Web Spam Filtering

We participated in the Web Spam Challenge contest in 2008. The goal of the Web Spam Challenge series is to identify and compare Machine Learning methods for automatically labeling websites as spam or normal. We made experiments with both MLDA and linked LDA. We ran our tests on the WEBSPAM-UK2007 dataset. The organizers provided baseline content and link based features, thus we were given a controlled environment to measure the performance of our models. In both cases, we applied feature combination schemes in order to improve classification performance. In the case of MLDA, we used a logistic regression based combination, whilst with linked LDA, we combined the results of the different classifiers by the random forest classifier.

Experimental results with MLDA

In MLDA, we build two separate LDA models, one for the spam and one for the normal web pages, then we take the union of the resulting topic collections and make inference with respect to this aggregated collection of topics for every unseen document d. Finally, the total probability of spam topics in the topic distribution of an unseen document gives a prediction of being spam or honest. The key results are as follows:

- We reached a relative improvement of 11% in *AUC* over the baseline classifiers with the MLDA model, and we ranked fourth in the competition with an $AUC = 0.80$.

The results appeared in the proceedings of the 4^{th} International Workshop on Adversarial Information Retrieval on the Web (*AIRWeb'08*).

Experimental results with linked LDA

In case of linked LDA, we simply apply our model to get the topic distribution of the Web pages, and then we perform spam classification. The key results are as follows:

- We tested linked LDA on the same corpus and we compared our results to the winner of 2008. Our random forest based combinations achieve an $AUC = 0.854$, while the winner's result was $AUC = 0.848$.

The results appeared in the proceedings of the 5^{th} International Workshop on Adversarial Information Retrieval on the Web (*AIRWeb'09*).

1.3 Organization

In Chapter 2, we introduce the problem of document modeling with focus on classification. Then we describe the core steps of a classification process: preprocessing, document representation, classifier training and evaluation.

The topic of Chapter 3 is LDA, the motivation of our research. Firstly, we start with an introduction to Bayesian Inference, the underlying mathematical method of LDA, then we discuss other latent variable models relevant to LDA. Finally, we describe the LDA modeling framework in detail.

In Chapter 4, we report our comparative analysis based on LSA and LDA, in a Web page classification application.

In Chapter 5, we introduce and evaluate a novel probabilistic topic exploration technique, called Multi-corpus Latent Dirichlet Allocation (MLDA), for supervised semantic categorization of Web pages.

Chapter 6 presents the applicability of LDA for classifying large Web document collections. One of our main results is linked LDA, a novel influence model that gives a fully generative model of the document content taking linkage into account. As another main contribution, we develop LDA specific boosting of Gibbs samplers resulting in a significant speedup in our experiments.

In Chapter 7 we present the results of our participation at Web Spam Challenge 2008 with our MLDA and linked LDA models.

Chapter 2

Document modeling and classification

We begin this chapter with an introduction to text mining and document classification. In the next four sections, we review the core steps of a typical document classification process: preprocessing, representation, classification and evaluation. In Section 2.3 we introduce the classification algorithms used in our experiments.

Text mining is a relatively new, interdisciplinary, application-oriented research area, first time mentioned in [39] and widely used from 2000. It uses similar techniques as *natural language processing* [74] and *information retrieval* [75], combining them with the algorithms and methods of *data mining* [52], *machine learning* [2] and *statistics*. Text Mining is the application of algorithms and methods from the fields of machine learning and statistics to text with the goal of exploring, identifying and analyzing implicit, previously unknown, and potentially useful information.

Although typical text mining tasks include classification [105], clustering [4], information extraction [103], automatic summarization [73], sentiment analysis [86], in this thesis we focus on **classification**.

The history of document classification dates back to Maron's [78] work on probabilistic text classification. Since then, it has been used for a number of different applications: automated document indexing [43, 114], automated metadata generation [51], document organization [68], word sense disambiguation [19, 102], text filtering [84], hierarchical categorization of web pages [36], etc.

The goal of document classification is to assign predefined classes to text documents [113]. As an example, we may automatically label each incoming news story with a topic like "science", "politics", or "culture". More formally, the task is to determine a *classification function*:

$$\psi : \mathcal{D} \rightarrow 2^{\mathcal{C}} \tag{2.0.1}$$

where $\mathcal{D} = \{d_i\}_{i=1}^{M}$ is the set of the documents and $\mathcal{C} = \{c_i\}_{i=1}^{K}$ is the set of pre-defined classes.

Function ψ is called the classifier. Thus the classifier assigns a subset of $\mathcal{C}$ to every document d_i.

Document classification has three types: *supervised, unsupervised* and *semi-supervised document classification*. In the first case, the classifier is trained on data previously labeled by human assessors. In the second case, the classification must be done without human assistance, sometimes even without the pre-defined label set. Finally, in the third case, the training document collection contains a large amount of unlabeled data together with labeled data. The semi-supervised classifier (under certain assumptions) could also learn from the unlabeled data. One of the main advantage of this methodology is:

"labels are hard to obtain while unlabeled data are abundant, therefore semi-supervised learning is a good idea to reduce human labor and improve accuracy" [126].

The classification task can be categorized based on the number of labels assigned to the documents: the general task described in (2.0.1) is the *multi-label* case, while at the *single-label* case, only one label is assigned to a document (hence the classifier is simply $\psi : \mathcal{D} \to \mathcal{C}$). There is also a special case in the single-label classification: the **binary classification**, where we have only two labels (positive and negative).

To build and evaluate a classifier in a typical supervised classification context, one has to do the following: given the category set $\mathcal{C}$ and a document collection (corpus) $\mathcal{D}$ assigned to $\mathcal{C}$, we split the corpus $\mathcal{D}$ into two disjunct parts; training $\mathcal{D}_{train}$ and test set $\mathcal{D}_{test}$. We build the classifier on $\mathcal{D}_{train}$, then we apply it to the documents from $\mathcal{D}_{test}$ and compare the original labels to the classifier output. As a specific implementation, in K*-fold cross validation* the document collection is partitioned into K subsamples. Of the K subsamples, a single subsample is retained for testing the classifier, and the remaining $K - 1$ subsamples are used as training data. The cross validation process is then repeated K times (the folds), with each of the K subsamples used exactly once as the testing data. The K results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method over repeated random sampling is that all observations are used for both training and testing, and each observation is used for testing exactly once. 10-fold cross validation is commonly used.

Sometimes it is important to adjust the input parameters of the classifier, for this purpose one has to hold out a $\mathcal{D}_{heldout}$ part of the training corpus. The input parameters can then be optimized in the $\{\mathcal{D}_{heldout}, \mathcal{D}_{test}\}$ pair, and evaluated on $\{\mathcal{D}_{train} \setminus \mathcal{D}_{heldout}, \mathcal{D}_{test}\}$. This technique can be applied to the cross validation scheme as well.

An important special case of document classification is **Web content classification**, a research area that abounds with opportunities for practical solutions. The performance of most traditional machine learning methods is limited by their disregard for the interconnection struc-

ture between Web data instances (nodes). At the same time, relational machine learning methods often do not scale to web-sized data sets.

The selection of an appropriate model for Web pages is crucial for implementing a Web page classifier. The goal is to obtain compact representations that are sufficiently descriptive and discriminative of the topics associated with the pages. Three important questions are (1) what features should be extracted from the pages, (2) how to use these features to model the content of each page, and (3) what algorithm should be used to issue a prediction of a page category.

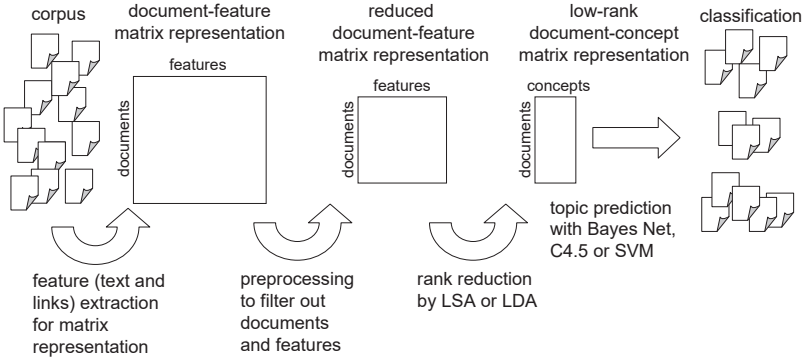

Figure 2.1: Steps of the Web page classification process.

Figure 2.1 summarizes the steps involved in our web classification experiments including the pre-processing of the corpus and the application of the classification algorithms.

Web page classification is different from traditional text classification because of the presence of links. While "text-only" approaches often provide a good indication of relatedness, the incorporation of link signals can considerably improve methods for grouping similar resources [6, 22, 95, 116].

A typical document classification process involves the following phases:

- preprocessing (Section 2.1)
- document representation (Section 2.2)
- classifier training (Section 2.3)
- evaluation (Section 2.4)

We briefly review these steps in the next four subsections.

2.1 Preprocessing

Because text cannot be directly interpreted by a classifier, preprocessing is needed to map text into a compact representation. The borderline between preprocessing techniques and document representation approaches are quite fuzzy, for example term selection or extraction can be considered as a preprocessing step or as a part of the representation. In our work, we follow the treatment of [60]. Text preprocessing usually consists of the following tasks:

1. text segmentation, tokenization,
2. stop word filtering,
3. stemming, lemmatization,
4. dimensionality reduction by term selection or extraction.

Text segmentation aims at breaking up a stream of text into meaningful linguistical units: words, phrases, sentences, paragraphs, chapters etc. *Tokenization* is the process of splitting a document into words (or tokens) by removing all punctuation marks and by replacing tabs and other non-text characters by single white spaces.

Stop words, e.g. prepositions, articles, conjunctions etc. have little information content. Additionally, words that occur extremely often have low potential to distinguish topics while very rare words are of no particular statistical relevance and can be removed from the document [7].

Stemming is the process for reducing inflected (or sometimes derived) words to their stem or root form, e.g. removing the plural 's' from nouns, the 'ing' from verbs, or other suffixes. After the stemming process, every word is represented by its stem. The most popular algorithm is the Porter stemmer [93], which is based on a set of transformation rules. However, it has several drawbacks: it can be used only for English[1] and causes over-stemming errors: the aggressive suffix removal often maps semantically different words to the same root, e.g. $designer \rightarrow design$ and $designate \rightarrow design$. These errors usually deteriorate the performance of IR systems [121]. *Lemmatization* captures a similar notion, although it applies a more complex two-level approach. In the first step we determine the *part-of-speech*[2] of a word, then we use part-of-speech dependent rules to find the correct root of the word. A popular algorithm is TreeTagger [104].

Term selection and *term extraction* refers to techniques with the goal to decrease the size of the original vocabulary from $|\mathcal{V}|$ to $|\widehat{\mathcal{V}}|$, where $|\widehat{\mathcal{V}}| \ll |\mathcal{V}|$.

Term selection can be achieved via filtering: we keep terms with highest score according to a function that measures the "importance" of the term. The most common importance measure

[1] Although it can be extended to process similar languages [92].
[2] Part-of-speech (POS) refers to the linguistic category of the word.

is the document frequency $df(t_k)$ of a term t_k. As a rule of thumb, the features can be reduced by a factor of 10 with no loss in effectiveness [122]. Extensive experiments were carried out with other term selection functions such as chi-square, information-gain, mutual information, etc.

During *term extraction*, the terms in $\widehat{\mathcal{V}}$ are obtained by combinations or transformations of the original ones. These synthetic terms can help to reduce the noise caused by polysemy, synonymy and homonymy. There are several term extraction approaches, e.g. term clustering, Latent Semantic Analysis (LSA), probabilistic Latent Semantic Analysis (PLSA), etc. These methods are described in detail in Section 2.2.

2.2 Text representation

There are numerous text representation approaches in the literature.We discuss selected methods used in this thesis and we refer to [65] for a more detailed survey.

2.2.1 Vector space model

The vector space model was introduced in [101]. We have a document collection $\mathcal{D} \equiv \{\vec{d_i}\}_{i=1}^{M}$. Documents are represented as finite dimensional vectors, that is $\vec{d_i} = (w_{i1}, \ldots, w_{iV})$, where $\mathcal{V}$ is the vocabulary and V is its size. The vocabulary contains terms occurring in at least one document of $\mathcal{D}$. The weight w_{ij} represents how much term t_j contributes to the content of the document $\vec{d_i}$.

The weights w_{ij} can be defined in several ways. In the simplest binary weighting 1 denotes presence and 0 denotes absence of the term in the document. The most widely used weighting is $tf \times idf$ [100]:

$$w_{ij} = n_{ij} \log \frac{M}{\left|\{\vec{d_k} : t_j \in \vec{d_k}\}\right|}, \tag{2.2.1}$$

where n_{ij} is the term frequency, or simply tf, the number of occurrences of term t_j in document $\vec{d_i}$. M is the number of documents and the denominator of the second term is the number of documents containing term t_j at least once. The second term of (2.2.1) is the inverse document frequency (or idf). This weighting embodies the intuition that the more often a term occurs in a document, the more it represents of its content, and the more documents a term occurs in, the less discriminating it is. There exist many variants of $tf \times idf$ that differ from each other in terms of logarithms or other correction factors. Definition (2.2.1) is just one of the possible instances of this class.

The similarity between two documents (or a document and a query) can be defined in several

ways using the $tf \times idf$ weights. The most common one is the cosine similarity:

$$sim(\vec{d_i}, \vec{d_j}) = \frac{\vec{d_i} \cdot \vec{d_j}}{\left\|\vec{d_i}\right\| \left\|\vec{d_j}\right\|}. \tag{2.2.2}$$

In a typical information retrieval context, given a query and a collection of documents, the task is to find the most relevant documents from the collection to the query. The first step is to calculate the vector space representation for the documents and the query, then to calculate the cosine similarity for those documents containing the query words and finally to select those having the highest similarity score.

Normalization is another important aspect of proper weighting. Document length normalization of term weights is used to adjust the overestimation of long documents in retrieval over the short documents. Two main reasons that necessitate the use of normalization in term weights are:

- Higher term frequencies: Long documents usually use the same terms repeatedly. As a result, term frequency factors may be large for long documents and this increases the average contribution of their terms toward the query-document similarity.
- More terms: Long documents also have numerous different terms. This increases the number of matches between a query and a long document, increasing the query-document similarity and the chances to retrieve long documents before short ones.

Document length normalization is a way of penalizing the term weights for a document in accordance with its length. Singhal in [108] examined the state of the art normalization schemes and proposed a pivoted document length normalization. It adapts the normalization to the document collection in context. Our experimental results showed significant improvements in the retrieval performance over the standard $tf \times idf$ weighting. The scheme is as follows:

$$w_{ij} = \frac{1 + \log(n_{ij})}{1 - s + s\frac{N_i}{\widehat{N}}} \log \frac{M}{\left|\{\vec{d_k} : t_j \in \vec{d_k}\}\right|} \tag{2.2.3}$$

where s is a constant (in practical uses 0.8), N_i is the length of $\vec{d_i}$ in terms of bytes instead of words, $\widehat{N}$ is the average of N_i over the corpus and the second term is the idf.

2.2.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA) introduced in 1988 by Deerwester et al. [31] is a method for extracting and representing the contextual meaning of words by statistical computations applied to a large corpus. The method uses the the vector space model with $tf \times idf$ term-weighting as input and does term extraction, see Subsection 2.1 for an overview.

An important step in LSA is to transform the term-document vector space into a concept-document and document-concept vector space. By reducing the number of concepts, the documents and their terms are projected into a lower-dimension concept space. As a consequence, new and previously latent relations will arise between documents and terms. In order to apply LSA we first generate a term-document matrix D from the given corpus. Then, singular-value decomposition (SVD) is applied to D. The SVD of the matrix D is defined as the product of three matrices,

$$D = U\Sigma V^T, \tag{2.2.4}$$

where the columns of U and V are the left and right singular vectors, respectively, corresponding to the diagonal elements of Σ called the singular values $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r$ of the matrix D. The columns of U and V are orthogonal so that $U^TU = V^TV = I_r$, where r is the rank of matrix D. The first k columns of U and V and $\sigma_1, \ldots, \sigma_k$ are used to construct a rank-k approximation to D via $D_k = U_k\Sigma_k V_k^T$. A theorem by Eckart and Young [117] claims that D_k is the closest rank-k approximation to D with respect to the Frobenius and spectral norms [9].

The rows of the reduced matrices, U' and V', respectively, are taken as coordinates of points representing the documents and terms in a k dimensional space. Using the SVD decomposition, one can compare two terms, two documents or a document with a term. For example a document-document similarity matrix can be defined as:

$$D_k^T D_k = V\Sigma^2 V^T. \tag{2.2.5}$$

The main advantages of LSA are:

- *Synonymy:* Synonymy refers to the fact that two or more different words have the same or similar meaning, such as *movie* and *film*. A traditional vector space model based IR system cannot retrieve documents discussing the topic of a given query unless they have common terms however after we map the query and the document to the concept space, they are both likely to be represented by a similar weighted combination of the SVD variables.

- *Polysemy:* Polysemy refers to the fact that one word have multiple meaning, such as the word *bank*. The precision of the retrieval can be reduced significantly, if the query have a large number of polysemous word. Applying LSA to the query we can filter out the rare and less important usages of certain terms, thereby increasing the precision of the search.

- *Term dependence*: The vector space model relies on the binary independence concept, i.e. the terms constituting the documents are completely independent from each other (they are orthogonal basis vectors of the vector space). However it is well known that there are strong correlations between terms. Term associations, for example can be exploited

by adding phrases composed of two or more words to the vocabulary. LSA offers a more intuitive solution through the embedding of word-word, document-document and word-document correlations into the reduced LSA factor based representation. Although it comes at the price of an increased computational cost, this is only a one-time cost because one can build the LSA representation for the entire document collection once (i.e. performance at retrieval time is not affected).

The main disadvantages of LSA are as follows:

- *LSA and normally-distributed data:* As noted earlier, SVD finds a k-dimensional approximation of the term-document matrix (say A_k), given that the Frobenius-norm of the error term ($A - A_k$) is minimized, or in other terms beside least-squares error. But, least-squares error is in fact suitable for normally-distributed data, not for count data as in the term-document matrix. A possible solution is to use $tf \times idf$ weighting before applying SVD [76].

- *Storage:* it seems to be antinomic, but the space requirement of an SVD representation for several real datasets is larger than the sparse representation [61].

- *Inefficient lookup:* using vector space representation, one can build an inverted index for the documents, i.e. a table where the keys are the words, and the contents are the documents containing the key-word. Consequently, only documents that have some terms in common with the query must be examined during the retrieval process. However in the LSA representation, the query must be compared to every document in the collection.

2.3 Classifier learning

In this section we briefly review the most common classification algorithms or learning schemes focusing on those which were used in our experiments. For a detailed explanation we refer the reader to [35, 105].

2.3.1 Bayesian Networks

Bayesian Networks are probabilistic graphical models in which nodes represent random variables and missing edges between the nodes represent conditional independence assumptions. If the edges are directed we call the network a Bayesian Network[3], while graphs with undirected edges belong to Markov Networks or Markov Random Fields.

[3]Belief network, generative model, causal model are popular alternate names.

Bayesian Networks will appear in two contexts in the present thesis. On the one hand, we use them as a general modeling framework since LDA is a hierarchical Bayes model represented by a 3 level Bayesian Network (more details in 3.4 Subsection). On the other hand, we also deploy them for supervised classification: given the LDA representation of documents and the corresponding category labels, one may apply classifiers from the Bayesian family (e.g. naïve Bayes, tree augmented naïve Bayes etc.) to classify the data. First we introduce the general framework, then we briefly discuss the naïve Bayes classifier and the BayesNet classifier used in the Weka machine learning toolkit [118].

Given a set of random variables $\{X_i\}_{i=1}^n$ and a Bayesian Network over them. The joint probability distribution $p(X_1, \ldots, X_n)$ is as follows:

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i | S(X_i)), \tag{2.3.1}$$

where $S(X_i)$ is the set of nodes in the graph that have a directed edge to X_i, also called the parents of X_i. Note, that it is possible that $S(X_i) = \emptyset$: in that case the local probability distribution of the node is said to be unconditional. A set of independent random variables are represented as nodes without edges between them, hence the (2.3.1) reduces to the well known factored representation of the joint probability distribution.

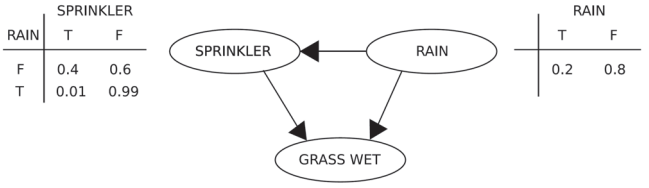

Figure 2.2: A simple Bayesian Network for modeling whether the grass is wet or not. The grass is wet if either the sprinkler is on or it is raining. The local probability distributions associated with a node are shown adjacent to the node.

The goal of learning a Bayesian Network is to determine both the structure of the network (*structure learning*) and all the conditional probability distributions (*parameter learning*).

Structure learning. The goal is to determine the structure of the network. Since the number of possible structures is extremely large, structure learning often has high computational complexity. Thus, heuristics and approximate learning algorithms are the realistic solutions. A variety of learning algorithms have been proposed [24, 26, 55]. One practical approach for structure learning is to impose some restrictions on the structure of the Bayesian Network, for example, learning tree-like structures, in which each node has at most one parent.

Parameter learning. In order to fully specify the Bayesian Network and thus the joint probability distribution, it is necessary to specify the probability distribution for X conditional upon X's parents for each node X. The distribution of X conditional upon its parents may have any functional form. If the conditional distributions contain parameters which are unknown and must be estimated from data, one resort to maximum likelihood estimation, MAP estimation or full Bayesian estimation, for a detailed explanation see Section 3.2. In case of presence of unobserved variables the most common approach to estimate the joint probability distribution is the expectation-maximization (EM) algorithm [32]. A simple Bayesian Network with the calculated probabilities is demonstrated in Figure 2.2, adopted from [99].

Naïve Bayes. A naïve Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naïve) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model". The network structure is in Figure 2.3.

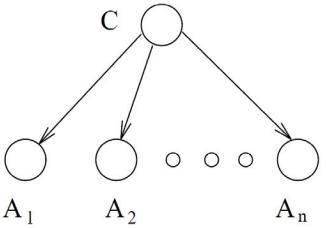

Figure 2.3: The structure of the naïve Bayesian Network. Features $A_1, \ldots, A_n$ are conditionally independent given the class label C.

Formally, for a document described by its terms or other features, $\vec{d} = (w_1, \ldots, w_i)$, the task is to estimate probabilities $p(C|\vec{d}\,) = p(C|w_1, \ldots, w_i)$. Using Bayes' rule we get:

$$p(C|w_1, \ldots, w_i) = \frac{p(C)p(w_1, \ldots, w_i|C)}{p(w_1, \ldots, w_i)} \tag{2.3.2}$$

From Bayesian terminology, the first term of the nominator is the *prior*, the second is the *likelihood*, and the left side of the equation is the *posterior*, see Section 3.2.1 for a detailed description. Applying the conditional independence assumption to the second term of the nominator, and considering that we are only interested in class $c \in C$, yielding the highest probability, we arrive at:

$$p(C|w_1, \ldots, w_i) \sim p(C) \prod_{l=1}^{i} p(w_l|C) \tag{2.3.3}$$

Often the prior probabilities $p(C)$ of all classes may be taken to be equal, thus only terms $p(w_l|C)$ should be learned from training data. Although this model is unrealistic due to its restrictive independence assumption, it yields surprisingly good classification accuracy [35, 62].

BayesNet in Weka. We use the BayesNet classifier implemented in Weka [118] with default settings in our experiments. For more information see [16].

2.3.2 Support Vector Machine

Support Vector Machines (SVM) are supervised learning methods that recently have been applied successfully to text classification tasks [35, 62, 66]. SVM is generally proposed for binary classification, however it can be extended to the multi-class case. The idea of SVM is based on the simple case when we can separate the positive and negative instances by a hyperplane. First we describe this simple setup, then we turn to the description of the SVM.

Let us suppose that the training documents are represented as V dimensional vectors, and two classes are given: positive and negative. In the case of a linear classifier, one would like to separate the documents with a $V-1$ dimensional hyperplane. From the possible separating hyperplanes we want to maximize their distance from the nearest data point. More formally:

$$\mathcal{D} = \left\{ (\vec{d_i}, c_i) \mid \vec{d_i} \in \mathbb{R}^V,\ c_i \in \{+1, -1\} \right\}_{i=1}^{m}, \tag{2.3.4}$$

where $\vec{d_i}$ represents the document and c_i its class label. We want to find the hyperplane[4] which divides the points with labels $c_i = +1$ from those having $c_i = -1$. Any hyperplane can be written as the set of points $\vec{d}$ satisfying

$$\vec{w}^T \cdot \vec{d} - \vec{b} = \vec{0}. \tag{2.3.5}$$

We want to choose the $\vec{w}$ and $\vec{b}$ to maximize distance between the parallel hyperplanes that are as far apart from each other as possible while still separating the data. It can be shown that these

[4]It is often called as maximum-margin hyperplane.

hyperplanes can be described by the equations

$$\begin{aligned} \vec{w}^T \cdot \vec{d} - \vec{b} &= +1 \\ \vec{w}^T \cdot \vec{d} - \vec{b} &= -1. \end{aligned} \tag{2.3.6}$$

Documents falling on these hyperplanes are the so called Support Vectors and the maximum margin hyperplane is at half-way between the Support Vector hyperplanes. This can be formulated as an optimization problem:

$$\begin{aligned} &\text{choose } \vec{w}, \vec{b} \quad \text{to minimize } \|\vec{w}\| \\ &\text{subject to } c_i(\vec{w}^T \vec{d_i} - \vec{b}) \geq 1, \text{for all } 1 \leq i \leq n \end{aligned} \tag{2.3.7}$$

After the above so-called linear SVM has been trained, the category of a point $\vec{y}$ can be calculated according to the formula:

$$c = \begin{cases} +1, \text{if} \quad \vec{w}^T \cdot \vec{y} - \vec{b} > 0 \\ -1, \text{if} \quad \vec{w}^T \cdot \vec{y} - \vec{b} < 0 \end{cases}$$

SVM can be applied when the training documents are not linearly separable in $\mathbb{R}^V$ and can be modified to create non-linear classifiers by using different kernel-functions in the vector inner-product. For a more thorough discussion of the topic can be found in [48].

SVM in Weka. We applied the Sequential Minimal Optimization (SMO) implementation in Weka [118] for training an SVM classifier with a linear kernel function and default values, see the details in [89].

2.3.3 C4.5 decision tree

Classifiers introduced in the previous sections behave like black boxes, i.e. they apply decision rules that are not easily interpretable by humans. On the contrary, decision tree based classifiers generate simple decision rules that can help to further analyse or to capture hidden associations among data instances. In this thesis, we only describe decision tree text classifiers: internal nodes in the tree represent terms from the given vocabulary, edges represent expressions testing the association of the given term with the test document, finally, leaves are labeled by categories. After the tree is constructed, it can be used for classifying a test document as follows: starting from the root, one recursively chooses a direction at all inner nodes based on evaluating the expressions on edges which in turn are based on the weight of the given term of the test document, until a leaf node is reached. The label of this node is then assigned to the document.

The most widely used classifiers use binary document representation (see Subsection 2.2.1), hence produce binary trees. An example of a decision tree built for the category *Wheat* in the Reuters-21578[5] collection is illustrated in Figure 2.4.

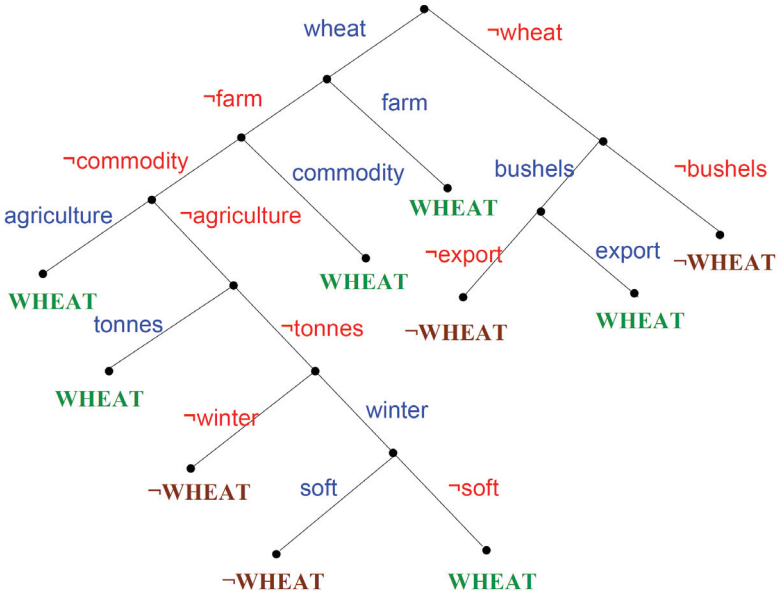

Figure 2.4: Decision tree example for the category *Wheat* from Reuters-21578. Edges are labeled by terms and leaves are labeled by the categories (¬ stands for negation).

Decision tree construction

The decision trees are usually constructed iteratively. The general step at a given node of the tree is to find the "best feature", then make a branch from this node for every value of the feature, then recursively repeat the branching process for the branches excluding the feature in the current node. The branches contain training data samples with the same feature value.

The state of the art algorithms differ in the definition of "best attribute" and the balance constraints on the tree. The most common algorithm is C4.5 developed by Ross Quinlan [97]. C4.5 is an extension of Quinlan's earlier Iterative Dichotomiser 3 (ID3) algorithm. The ID3 algorithm uses the information gain to select the "best feature", for further details see [96].

[5] http://daviddlewis.com/resources/testcollections/reuters21578/

C4.5 in Weka. We applied the C4.5 implementation in Weka with default values, for more information see [97].

2.3.4 Stacked graphical learning

Stacked graphical learning is a general learning scheme developed by Kou and Cohen [63]. As a relational learning method, it exploits both labeled and unlabeled training instances to build a classifier, relying on the interconnection structure behind the data (e.g. the hyperlinked environment of documents).

General scheme

Given the training data composed of labeled (for the sake of simplicity, with only positive or negative labels) and unlabeled instances. The general stacked graphical procedure starts with one of the base learners introduced in Subsections 2.3.1, 2.3.2, 2.3.3 that classifies each element v with weight $p(v) \in [0, 1]$. Positive and negative instances in the training set have $p(v)$ equal 0 and 1, respectively. These values are used in a classifier stacking step to form new features $f(u)$ based on $p(v)$ of the neighbors v of u. Finally the base learner is applied to the expanded feature sets to make the final predictions. There are some technical details in connection with the calculation of $f(u)$.

- How can one propagate the information between nodes?
- How can one define the similarities between nodes (what kind of edge-weight function is used)?
- What is the optimal distance to collect information from the neighbors?

Focusing on the problem of Web page classification, the possible solutions could be as follows.

Direction of propagation

We may use both the input directed graph, its transpose by changing the direction of each edge, or the undirected version arising as the union of the previous two graphs. We will refer to the three variants as *directed*, *reversed* and *undirected* versions. For an edge weight function $d : V \times V \to \mathbf{R}$ we use $d^-(u, v) = d(v, u)$ for the reversed and $d^\pm = d + d^-$ for the undirected version. We extend this notion for an arbitrary similarity measure $\mathrm{sim}(u, v)$ computed over edge weights d and compute $\mathrm{sim}^-(u, v)$ over d^- and $\mathrm{sim}^\pm(u, v)$ over $d^\pm$.

Edge-weight functions

Cocitation $\text{coc}(u, v)$ is defined as the number of common inneighbors of u and v. This measure turned out to be most effective for Web spam classification [8]. Using the notation of Section 2.3.4, $\text{coc}^-(u, v)$ denotes bibliographic coupling (nodes pointed to by both u and v) and $\text{coc}^\pm(u, v)$ is the undirected cocitation. We may also use cocitation downweighted by degree, $\text{coc}(u, v)/d(u) \cdot d(v)$.

The Jaccard and cosine similarity coefficients are useful for finding important connections and ignoring "accidental" unimportant connections. The Jaccard coefficient $\text{Jac}(u, v)$ is the ratio of common neighbors within all neighbors. The coefficient has variants that use the reversed or undirected graphs. For a weighted graph we may divide the total weight to common neighbors by the total weight of edges from u and v. This measure performs poor if for example edges ux and vy have low weight while uy and vx have very high, since the Jaccard coefficient is high while the actual similarity is very low.

Cosine similarity fixes the above problem of the Jaccard coefficient. We consider the row of the adjacency matrix corresponding to node u as vector $\overline{u}$. The cosine similarity of nodes u and v is simply $\cos(u, v) = \overline{u}^T\overline{v}$. We may similarly define $\cos^-(u, v)$ over the transpose matrix and $\cos^\pm(u, v)$ over the sum.

Several other options to measure node similarities and form the neighborhood aggregate features are explored in [8, 28]. In Chapter 6 we propose a linked LDA model, which naturally produce an edge-weight function that can also be used in a stacking scheme with very good results.

2.4 Evaluation

In this section we define the most commonly used evaluation measures in classification, namely the *accuracy* (*ACC*), *precision (PR)*, *recall (R)*, *F-measure (F)* and *Area Under Curve (AUC)*.

Accuracy is the fraction of correctly classified documents among all documents. Precision is the fraction of retrieved documents that are relevant, i.e. belong to the target class, while recall gives the fraction of the relevant documents retrieved. These measures can also be defined in terms of a contingency table with the content of the true labels and the labels given by the classifier, see Table 2.1.

FP (false positives) is the number of negative instances classified incorrectly as positive, *FN* (false negatives) is the number of incorrectly classified positive instances, while *TP* (true positives) and *TN* (true negatives) are the number of documents classified correctly as positive and negative, respectively. In terms of the entries of the contingency table, the previous measures

		True labels	
		+	-
Classifier labels	+	*TP*	*FP*
	-	*FN*	*TN*

Table 2.1: Contingency table

are defined as follows:

$$ACC = \frac{TP + TN}{P + N} \tag{2.4.1}$$

where *ACC* is the accuracy and *P* and *N* refers to the number of positive and negative documents, respectively.

$$PR = \frac{TP}{TP + FP} \quad \text{and} \quad R = \frac{TP}{TP + FN} \tag{2.4.2}$$

where *PR* and *R* are the precision and recall.

Obviously there is a trade-off between precision and recall. Precision can be increased if we assign positive labels to only those documents of which we are really certain that they belong to the positive class, thereby reducing the fraction of true positives may result in a lower recall. On the other hand, if we accept more and more documents as positive, the fraction of false negatives reduces resulting in higher recall and decreased precision.

A compromise between precision and recall is the *F*-measure, the weighted harmonic mean of precision and recall:

$$F = \frac{2}{\frac{1}{R} + \frac{1}{PR}} \tag{2.4.3}$$

The *AUC* measure is defined over the Receiver Operating Characteristics (ROC) graph from signal detection theory. The ROC graph is a technique for visualizing, organizing and selecting classifiers based on their performance. First we define the true positive rate (*TPR*) and the false positive rate (*FPR*), from table 2.1 as

$$TPR = \frac{TP}{P} \quad \text{and} \quad FPR = \frac{FP}{N}. \tag{2.4.4}$$

The ROC graph is a two-dimensional graph in which *TPR* is plotted as the function of *FPR*. A discrete classifier is one that outputs only a class label. Each discrete classifier produces an (TPR, FPR) pair corresponding to a single point in ROC space. However, probabilistic classifiers[6], such as C4.5 or SVM naturally yield a probability or score which represents the degree to which an instance is a member of a class. Such a classifier can be used with a threshold to produce a discrete (binary) classifier: if the classifier output is above the threshold, the classifier

[6]We use the term probabilistic classifier in spite of the fact that the output may not be a proper probability, although the score can always be converted into a probability measure.

produces a $+$, otherwise a $-$. Each threshold value produces a different point in ROC space that result in the ROC curve. For comparing classifiers, however, we may want to reduce ROC performance to a single scalar value representing expected performance. A common method is to calculate the area under the ROC curve, abbreviated as *AUC* [17]. *AUC* is a portion of the area of the unit square, hence $AUC \in [0, 1]$. However, because random guessing produces the diagonal line between $(0, 0)$ and $(1, 1)$, with an area of 0.5, no meaningful classifier should have an *AUC* less than 0.5. *AUC* has an important statistical property: the *AUC* of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. This is equivalent to the Wilcoxon test of ranks [53].

So far we have considered the binary class case, although *AUC* can be straightforwardly generalized to the multi-class case. ROC curve for class c_i plots the classification performance using class c_i as the positive class and all other classes as the negative class, i.e. $P_i = c_i$ and $N_i = \bigcup_{j \neq i} c_j$. From the $|C|$ different AUC_i values, the total *AUC* can be calculated as

$$AUC = \sum_{c_i \in C} AUC_i \cdot p(c_i) \tag{2.4.5}$$

where $p(c_i)$ is the prevalence of the class c_i, i.e. the ratio of documents labeled as c_i.

Chapter 3

Latent Dirichlet Allocation

In this chapter we discuss Latent Dirichlet Allocation, one of the most successful generative latent topic models developed by Blei, Ng and Jordan [13]. As a warm up, assume that we want to model how a collection of documents is generated. Documents are represented as mixtures of topics, e.g. an article about trading with electron microscopes is composed of **business** and **science**, while topics are represented as mixtures of words, e.g. in the topic closely related to "computer science" words like *operating system, hdd, algorithm* have large frequency, while words like *checkmate, Heineken, carving* have small. An illustrative example can be seen in Figure 3.1.

Latent Dirichlet Allocation (LDA) models the generation of words in the documents as follows: at the beginning all the documents are empty, but we know the topic mixture of all the documents and the word mixture of all the topics. For each document, we choose a topic z from the document specific topic mixture, then we choose a word w from the mixture of z, then we repeat this process until we fill all of the word positions in the given document. Then, we restart the process with the next document, and so on.

In reality, we have exactly the opposite information, namely we know the words of the documents, and we would like to estimate the topic and word mixtures, respectively. This process is called *inference*.

First of all we review the historical background and evolution of LDA, then we introduce the Bayesian inference in Section 3.2. In Section 3.3, we survey other latent variable models, finally, in Section 3.4 we give a thorough treatment of LDA.

3.1 History of LDA

The main motivation behind the development of LDA is to find a "suitable" modeling framework for text corpora, or any other kind of discrete data. A primary information retrieval goal is to find a low dimensional representation that supports efficient operations such as document

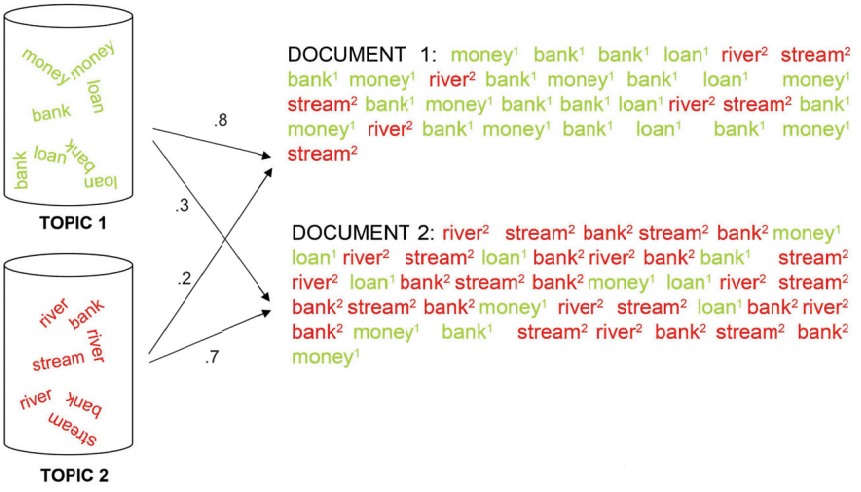

Figure 3.1: LDA example from [70]. Barrels represent topics as "bag of words" and arrows refer to the probability of choosing the given topic. There are two topics, **TOPIC**$_1$ captures **business**, while **TOPIC**$_2$ captures **geography**. *Document 1* discusses more **business** and less **geography**, while *Document 2* is more about **geography** and less about **business**.

retrieval, classification, clustering, etc., while keeping the inter- or intra-document statistical relationship as much as possible. First we review the three most influential approaches in this field.

The $tf \times idf$ scheme models every document with real valued vectors. The weights represent the contribution to the content of the document. It has been proved that this representation reveals little in the way of inter- or intra-document statistical structure, and the goal of reducing the description length (representation) of documents is only mildly alleviated. Details can be found in Subsection 2.2.1.

Latent Semantic Analysis achieves significant compression. In LSA, every document is rep-

resented with k-dimensional vectors, where k is significantly smaller then the length of the document, and these features can explain synonymy and polysemy, see the details in Subsection 2.2.2.

Probabilistic Latent Semantic Analysis embeds the notion of LSA into a probabilistical framework. It models every document with a topic-mixture. This mixture is assigned to the documents individually, without a generative process, the mixture weights are learned by expectation maximization.

Both LSA and PLSA rely on the "bag of words" assumption, that is the order of words is irrelevant. In probability theory, this is equivalent to the exchangeability property.

Definition 1 (Exchangeability) *A finite set of random variables* $\{X_1, \ldots, X_k\}$ *is exchangeable if*

$$p(X_1, \ldots, X_k) = p(X_{\pi(1)}, \ldots, X_{\pi(k)}), \tag{3.1.1}$$

that is the joint distribution is invariant to permutation. Here, $\pi(\cdot)$ *is a permutation function on the integers* $\{1 \ldots k\}$.

According to the classical representation theorem of de Finetti [29], a set of exchangeable random variables always have a representation as a mixture distribution. Note that an assumption of exchangeability is weaker than the assumption that the random variables are independent and identically distributed. Rather, exchangeability essentially can be interpreted as meaning "conditionally independent and identically distributed", where the conditioning is with respect to an underlying latent parameter of a probability distribution. The conditional probability distribution $p(X_1, \ldots, X_k|\Theta) = \prod_{i=1}^{k} p(X_i|\Theta)$ is easy to express, while the joint distribution usually cannot be decomposed.

This property was exploited in PLSA only on the level of words. As one can argue that not only words but also documents are exchangeable, a sound probabilistic model must capture the exchangeability of both words and documents. The LDA is a model which takes this issue into consideration.

3.2 Bayesian Inference

In LDA, Bayesian methods are usually applied for estimating the model parameters. In this section we briefly review the theoretical background of Bayesian estimation and the related concepts.

Bayesian inference is a general statistical inference method based on Bayes' Theorem. Given some evidence we would like to infer the probability of whether a hypothesis is true

or not. The process can be repeated if new evidence is observed and then the former probability can be updated. For example in *parameter estimation*, one is given a data set $\chi \equiv \{x_i\}_{i=1}^{|\chi|}$, which can be considered as a sequence of independent and identically distributed (i.i.d.) realizations of a random variable X. This random variable has a parametric probability distribution $p(X|\vartheta)$. The goal is to estimate parameters ϑ of X that can best explain the observation data χ. By Bayes' Theorem:

$$p(\vartheta|\chi) = \frac{p(\chi|\vartheta)p(\vartheta)}{p(\chi)}, \tag{3.2.1}$$

where $p(\chi|\vartheta)$ is the likelihood function, $p(\vartheta)$ is the prior, $p(\chi)$ is the evidence and $p(\vartheta|\chi)$ is the posterior distribution. Following the terminology:

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}. \tag{3.2.2}$$

The prior probability distribution $p(\vartheta)$ expresses uncertainty about ϑ before the observation data is taken into account. The posterior probability $p(\vartheta|\chi)$ is the conditional probability of the variable taking the evidence into account. The posterior probability is computed from the prior and the likelihood function via Bayes' theorem as shown in (3.2.2). The likelihood function $p(\chi|\vartheta)$, or the likelihood is a function of the parameters of a statistical model, that is, a conditional probability function considered as a function of its second argument with its first argument held fixed.

Now we show different estimation methods that start from simple maximization of the likelihood, then we show how prior belief on parameters can be incorporated by maximizing the posterior, and finally we use Bayes' rule to infer a complete posterior distribution.

Maximum likelihood estimation tries to find parameters that maximize the likelihood:

$$\widehat{\vartheta}_{ML} = \underset{\vartheta}{\operatorname{argmax}}\, p(\chi|\vartheta) = \underset{\vartheta}{\operatorname{argmax}} \sum_{x\in\chi} \log p(x|\vartheta) \tag{3.2.3}$$

The second equation is valid, because one assumes that the data drawn from a particular distribution is independent, identically distributed (iid) with unknown parameters. Thus the likelihood function can be written as $p(\chi|\vartheta) = \prod_{x\in\chi} p(x|\vartheta)$, and maxima is unaffected by monotone transformations. The solution for (3.2.3) can be found numerically by using various optimization algorithms, for example Lagrange multipliers, hill-climbing, simulated annealing etc.

Maximum a posteriori (MAP) estimation is very similar to maximum likelihood estimation but allows to include some a priori belief on the parameters by weighting them with a prior distribution $p(\vartheta)$. The name derives from the objective to maximize the posterior of the

parameters given the data:

$$\begin{aligned}\hat{\vartheta}_{MAP} &= \underset{\vartheta}{\operatorname{argmax}}\, p(\vartheta|\chi) = \underset{\vartheta}{\operatorname{argmax}}\, \frac{p(\chi|\vartheta)p(\vartheta)}{p(\chi)} = \underset{\vartheta}{\operatorname{argmax}}\, p(\chi|\vartheta)p(\vartheta) = \\ &= \underset{\vartheta}{\operatorname{argmax}} \left\{ \sum_{x \in \chi} \log p(x|\vartheta) + \log p(\vartheta) \right\} \end{aligned} \tag{3.2.4}$$

Compared to (3.2.3), a prior distribution is added to the likelihood. In practice, the prior $p(\vartheta)$ can be used to encode extra knowledge as well as to prevent overfitting by enforcing preference to simpler models. The MAP estimation in (3.2.4) can be computed via numerical optimization, Monte Carlo simulation, using a modified expectation-maximization, or analytically, if the distributions obeys certain conditions. With the incorporation of $p(\vartheta)$, MAP follows the Bayesian approach to data modeling where the parameters ϑ are thought of as random variables. With priors that are parametrized themselves, that is $p(\vartheta) := p(\vartheta|\alpha)$ with hyper-parameters α, the belief in the anticipated values of ϑ can be expressed within the framework of probability[1], and a hierarchy of parameters can be created.

Full Bayes estimation extends the MAP approach by allowing a distribution over the parameter set ϑ instead of making a direct estimate. In that case one has to calculate the normalization term in Equation 3.2.1, i.e. the probability of the "evidence". Its value can be expressed by:

$$p(\chi) = \int_{\vartheta \in \Theta} p(\chi|\vartheta)p(\vartheta)d\vartheta. \tag{3.2.5}$$

Using Bayes' Theorem, the full posterior distribution can be calculated. The intricate part of the full Bayesian approach is the calculation of the above mentioned normalization integral. Calculation of Bayesian models often becomes quite difficult, e.g., because the summations or integrals of the marginal likelihood are intractable or there are unknown variables. Fortunately, the Bayesian approach leaves some freedom to the encoding of prior belief, and a frequent strategy to facilitate model inference is to use *conjugate prior distributions*. A class of prior probability distributions $p(\vartheta)$ is said to be conjugate to a class of likelihood functions $p(x|\vartheta)$ if the resulting posterior distributions $p(\vartheta|x)$ are in the same family as $p(\vartheta)$. For example, the Gaussian family is conjugate to itself, the beta-distribution is conjugate to the Bernoulli or to the binomial distribution and in the multidimensional case, the Dirichlet distribution is conjugate to the multinomial distribution.

[1]Belief is not identical to probability, which is one of the reasons why Bayesian approaches are disputed by some theorists despite their practical importance [56].

3.2.1 Distributions

In this section we discuss probability distributions relevant to text modeling. In the LDA modeling framework, the multinomial and its conjugate prior, the Dirichlet distribution will be heavily used. These distributions are the k-dimensional generalized versions of the binomial and beta distributions, respectively. We will also need the definitions of the gamma function, and the $k-1$-dimensional simplex in $\mathbb{R}^k$.

Definition 2 (Binomial distribution) *The probability of getting k successes in a sequence of n independent true/false experiments. If $X \sim B(n,p)$, then $p(X=k) = \binom{n}{k} p^k (1-p)^{n-k}$.*

Definition 3 (Multinomial distribution) *The multivariate generalization of the binomial distribution. The experiment has k possible outcomes with probabilities $\mathbf{p} = (p_1, \ldots, p_k)$. Let the random variables X_i indicate the number of times outcome number i was observed over the n trials, then $X = (X_1, \ldots, X_k)$ follows a multinomial distribution with parameters n and $\mathbf{p}$ and*

$$p(\mathbf{X} = \mathbf{x}) = p(X_1 = x_1, \ldots, X_k = x_k) = \frac{n!}{x_1! \cdots x_k!} p_1^{x_1} \cdots p_k^{x_k} \tag{3.2.6}$$

Definition 4 (Beta-distribution)

$$p(\theta|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1} \tag{3.2.7}$$

where $\theta \in [0,1]$, and $\Gamma(x)$ is the Gamma function, where $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt$, and $\Gamma(n) = (n-1)!$ for a natural number n.

Definition 5 (($k-1$)-simplex)

$$\Delta^{k-1} = \left\{ (x_1, \cdots, x_k) \in \mathbb{R}^k \mid \Sigma_i x_i = 1 \text{ and } x_i \geq 0 \text{ for all } i \right\} \tag{3.2.8}$$

Definition 6 (Dirichlet-distribution) *A k-dimensional Dirichlet-distribution is defined as:*

$$p(\vec{\theta}|\vec{\alpha}) \propto Dir(\vec{\alpha}) = \frac{\Gamma\left(\sum_{i=1}^k \alpha_i\right)}{\prod_{i=1}^k \Gamma(\alpha_i)} \prod_{i=1}^k \theta_i^{\alpha_i - 1} = \frac{1}{\Delta(\vec{\alpha})} \prod_{i=1}^k \theta_i^{\alpha_i - 1}, \tag{3.2.9}$$

where the parameter $\vec{\alpha}$ is a k-dimensional vector with components $\alpha_i \in R^+$, and $\Gamma(x)$ is the Gamma function. A k-dimensional Dirichlet random variable $\vec{\theta}$ is embedded in the $(k-1)$-simplex. This is a conjugate prior distribution to the multinomial, which property will facilitate the Bayesian inference (see details in page 31). It can be shown that the function $\Delta(\vec{\alpha})$ is equal to the Dirichlet integral of the first kind for the summation function:

$$\Delta(\vec{\alpha}) = \int_{\sum x_i = 1} \prod_i^N x_i^{\alpha_i - 1} \mathrm{d}^N \boldsymbol{x}. \tag{3.2.10}$$

In figure 3.2 one can see the geometric interpretation.

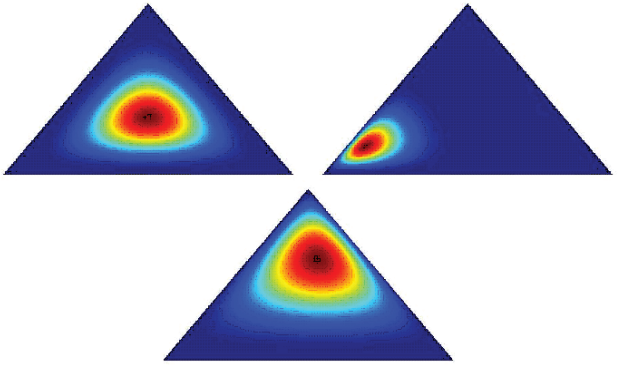

Figure 3.2: 3-dimensional Dirichlet-distributions with different $\vec{\alpha}$ parameters. In the first case $\vec{\alpha} = (4, 4, 4)$, in the second $\vec{\alpha} = (8, 1, 1)$, while in the third $\vec{\alpha} = (1, 4, 1)$. Colors with more red indicate larger probabilities.

3.3 Latent variable models

Next, we discuss simpler latent variable models: the *unigram model*, a *mixture of unigrams*, and the *PLSA model* from the point of Bayesian Networks (see Subsection 2.3.1 for a more thorough discussion).

The notation in Bayesian Networks is as follows. Circles correspond to hidden random variables, while double circles denote an evidence node, i.e. an observed variable. Conditionally independent replications of nodes are represented by plates (or rectangular boxes); note that this is equivalent to exchangeability, which is the property of a set of nodes to be invariant to permutation. Arrows refer to conditional probability distributions from the conditioned to the conditional variable.

Unigram model

Under the unigram model, the words $w_{m,n}$ of every document $d_m \equiv \vec{w}_m = \{w_{m,n}\}_{n=1}^{N_m}$ are drawn independently from a single multinomial distribution $\vec{\varphi}$. This is illustrated in the graphi-

cal model in Figure 3.3. The probability of a document is as follows

$$p(\vec{w}_m|\vec{\varphi}) = \prod_{n=1}^{N_m} p(w_{m,n}|\vec{\varphi}). \tag{3.3.1}$$

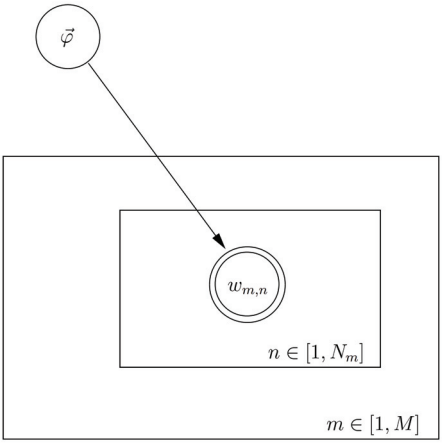

Figure 3.3: Unigram model represented as a graphical model.

Mixture of unigrams

In the mixture of unigrams model, each document d_m is generated by first choosing a topic indicator $z_m \in [1, K]$ and then generating N_m words independently from the corresponding conditional multinomial $\vec{\varphi}_{z_m}$. The graphical model of mixture of unigrams is in Figure 3.4. The probability of a document is:

$$p(\vec{w}_m|\underline{\Phi}) = \sum_{z_m} \prod_{n=1}^{N_m} p(w_{m,n}|\vec{\varphi}_{z_m}) p(z_m|\vec{\vartheta}_m), \tag{3.3.2}$$

where $\underline{\Phi} = \{\vec{\varphi}_k\}_{k=1}^{K}$ are multinomial distributions. These can be viewed as representations of topics and the documents are modeled as discussing only single topics. However, as the empirical results in [13] illustrate, this assumption is often too limiting to effectively model a large collection of documents.

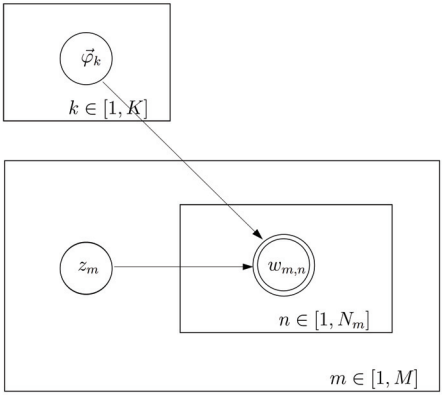

Figure 3.4: Mixture of unigrams model represented as a graphical model.

PLSA model

Probabilistic Latent Semantic Analysis (PLSA), or Probabilistic Latent Semantic Indexing (PLSI, in the field of information retrieval) is a latent topic mixture model that can be used for analyzing discrete data (e.g. a document collection with vector space representation). As noted in the introductory section of this chapter, LDA evolved from PLSA by extending the exchangeability property to the level of documents by applying Dirichlet priors on the multinomial distributions $\vec{\vartheta}_m$ and $\vec{\varphi}_k$, thus we are giving a more thorough presentation of this model.

The pseudo-code of the generative process of PLSA can be seen in Algorithm 3.3.1 and the graphical model is in Figure 3.5. Note that the difference between the mixture of unigrams and PLSA is the topic indicator variable $z_{m,n}$, which is re-sampled for each word.

Algorithm 3.3.1 Generative process for PLSA

1: "document plate"
2: **for all** documents $m \in [1, M]$ **do**
3: "word plate"
4: **for all** words $n \in [1, N_m]$ in document m **do**
5: sample topic index $z_{m,n} \sim Multinomial(\vec{\vartheta}_m)$
6: sample term for word $w_{m,n} \sim Multinomial(\vec{\varphi}_{z_{m,n}})$
7: **return**

According to the graphical model, one can infer that a document d_m and a word $w_{m,n}$ are conditionally independent given a hidden topic $z_{m,n}$, thus the probability of a document d_m is

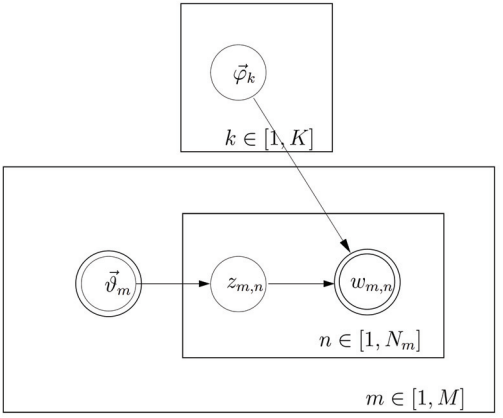

Figure 3.5: PLSA model represented as a graphical model.

as follows:

$$p(\vec{w}_m|\vec{\vartheta}_m, \underline{\Phi}) = \prod_{n=1}^{N_m} \sum_{z_{m,n}} p(w_{m,n}|\vec{\varphi}_{z_{m,n}}) p(z_{m,n}|\vec{\vartheta}_m). \tag{3.3.3}$$

PLSA was influenced by LSA (for further details see Subsection 2.2.2), as a solution for the unsatisfactory statistical foundation [59]: the main difference between LSA and PLSA is the objective function; in LSA, this is the L_2 or Frobenius norm, while in PLSA it is the Kullback-Leibler divergence between the model and the empirical data. As noted in the disadvantages of LSA in Section 2.2.2, the Frobenius norm is suitable for Gaussian data, which hardly holds for count data. The directions of the LSA latent space has no interpretable meaning, but those in the PLSA space can be interpreted as multinomial word distributions.

Model estimation. According to the likelihood principle, one determines the model parameters, $p(\vec{\vartheta}_m)$, $p(w_{m,n}|z_{m,n})$ and $p(z_{m,n}|\vec{\vartheta}_m)$ by maximization of the log-likelihood of the complete corpus. The standard procedure for maximum likelihood estimation in latent variable models is the expectation maximization (EM) algorithm [32]. EM alternates in two steps: (i) an expectation (E) step where posterior probabilities are computed for the latent variables, (ii) a maximization (M) step, where parameters are updated. In order to avoid overfitting, the authors proposed a widely applicable generalization of maximum likelihood model fitting by tempered

EM, which is based on entropic regularization and is closely related to deterministic annealing [98].

3.4 LDA modeling framework

3.4.1 Bayesian Network of LDA

The Bayesian Network of LDA can be seen in Figure 3.6. There are three plates, a document plate, a word plate and a topic plate with M,N_m, K replication counts, respectively. The model has three levels: $\vec{\alpha}$, $\vec{\beta}$, $\vec{\varphi}_k$ parameters are corpus-level parameters, assumed to be sampled once in the process of generating the corpus. The variables $\vec{\vartheta}_m$ are document-level variables, sampled once per document. Finally, the variables $z_{m,n}$ and $w_{m,n}$ are word-level variables and are sampled once for each word in each document. The complete generative model is summarized in Algorithm 3.4.1 and Figure 3.7 gives a list of all involved quantities.

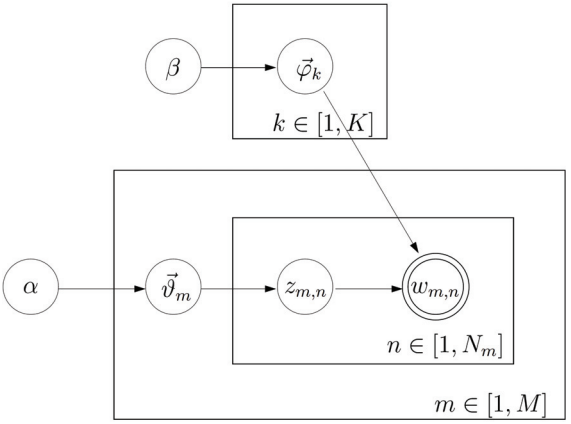

Figure 3.6: LDA as a Bayesian Network. The boxes are "plates" representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document.

Given the LDA parameters one can express various probability distributions, such as the probability that a word $w_{m,n}$ instantiates a term:

$$p(w_{m,n}|\vec{\vartheta}_m, \underline{\Phi}) = \sum_{z_{m,n}} p(w_{m,n}|\vec{\varphi}_{z_{m,n}}) p(z_{m,n}|\vec{\vartheta}_m). \tag{3.4.1}$$

Algorithm 3.4.1 Generative model for LDA from [56]

1: "topic plate"
2: **for all** topics $k \in [1, K]$ **do**
3: sample mixture components $\vec{\varphi}_k \sim Dir(\vec{\beta})$
4: "document plate"
5: **for all** documents $m \in [1, M]$ **do**
6: sample mixture proportion $\vec{\vartheta}_m \sim Dir(\vec{\alpha})$
7: "word plate"
8: **for all** words $n \in [1, N_m]$ in document m **do**
9: sample topic index $z_{m,n} \sim Multinomial(\vec{\vartheta}_m)$
10: sample term for word $w_{m,n} \sim Multinomial(\vec{\varphi}_{z_{m,n}})$
11: **return**

M	number of documents in the corpus
K	number of latent topics/ mixture components
V	number of terms t in vocabulary
N	number of words in the corpus, i.e. $N = \sum_{m=1}^{M} N_m$
$\vec{\alpha}$	hyperparameter on the mixing proportions
$\vec{\beta}$	hyperparameter on the mixture components
$\vec{\vartheta}_m$	parameter notation for $p(z\|d = m)$, the mixture component of topic k. One component for each topic, $\underline{\Theta} = \{\vec{\vartheta}_m\}_{m=1}^{M}$
$\vec{\varphi}_k$	parameter notation for $p(t\|z = k)$, the topic mixture proportion for document m. One proportion for each document, $\underline{\Phi} = \{\vec{\varphi}_k\}_{k=1}^{K}$
N_m	document length (document-specific), modeled with a Poisson distribution with constant parameter ξ
$z_{m,n}$	mixture indicator that chooses the topic for the n^{th} word in document m
$w_{m,n}$	term indicator for the n^{th} word in document m

Figure 3.7: Quantities in the model of LDA.

The complete-data likelihood of a document, the joint distribution of all known and hidden variables given the hyperparameters can be expressed as:

$$p(\vec{w}_m, \vec{z}_m, \vec{\vartheta}_m, \underline{\Phi}|\vec{\alpha}, \vec{\beta}) = \overbrace{\underbrace{\prod_{n=1}^{N_m} p(w_{m,n}|\vec{\varphi}_{z_{m,n}}) p(z_{m,n}|\vec{\vartheta}_m)}_{\text{word plate}} \cdot p(\vec{\vartheta}_m|\vec{\alpha})}^{\text{document plate (1 document)}} \cdot \underbrace{p(\underline{\Phi}|\vec{\beta})}_{\text{topic plate}} . \tag{3.4.2}$$

By integrating out the distributions $\vec{\vartheta}_m$ and $\underline{\Phi}$ and summing over $z_{m,n}$, one can get the likelihood of a document $\vec{w}_m$, i.e., of the joint event of all words occurring:

$$p(\vec{w}_m|\vec{\alpha},\vec{\beta}) = \iint p(\vec{\vartheta}_m|\vec{\alpha}) \cdot p(\underline{\Phi}|\vec{\beta}) \prod_{n=1}^{N_m} \sum_{z_{m,n}} p(w_{m,n}|\vec{\varphi}_{z_{m,n}}) p(z_{m,n}|\vec{\vartheta}_m) \, \mathrm{d}\underline{\Phi} \, \mathrm{d}\vec{\vartheta}_m$$
$$= \iint p(\vec{\vartheta}_m|\vec{\alpha}) \cdot p(\underline{\Phi}|\vec{\beta}) \prod_{n=1}^{N_m} p(w_{m,n}|\vec{\vartheta}_m, \underline{\Phi}) \, \mathrm{d}\underline{\Phi} \, \mathrm{d}\vec{\vartheta}_m. \tag{3.4.3}$$

Finally, the complete corpus $\mathcal{W} = \{\vec{w}_m\}_{m=1}^{M}$ likelihood is equal to the product of the likelihood of the independent documents:

$$p(\mathcal{W}|\vec{\alpha},\vec{\beta}) = \prod_{m=1}^{M} p(\vec{w}_m|\vec{\alpha},\vec{\beta}). \tag{3.4.4}$$

Comparison of LDA and PLSA

The most important difference between the two models is how the document topic mixtures are modeled. In PLSA $\vec{\vartheta}_m$ is a multinomial random variable with as many values as training documents. The model learns these topic mixtures only for the training documents, thus PLSA is not a fully generative model, particularly at the level of documents, because there is no straightforward solution to assign probability to a previously unseen document. As another consequence, the number of parameters which must be estimated grows linearly with the number of training documents. The parameters for a k-topic PLSA model are k multinomial distributions of size V and M mixtures over the k hidden topics. This gives $kV + kM$ parameters and therefore grows linearly in M. The linear growth in parameters suggests that the model is prone to overfitting. Although the authors of PLSA proposed a tempering heuristic to smooth the parameters of the model for acceptable predictive performance, it has been shown that overfitting can occur even when tempering is used [90].

LDA overcomes both of these problems by treating the topic mixture weights as a k-parameter hidden random variable sampled from a Dirichlet distribution, hence sampling for unseen documents (e.g. test documents) can be easily done. Furthermore, the number of parameters is $k + kV$ in a k-topic LDA model, which do not grow with the size of the training corpus, thereby avoids overfitting.

3.4.2 Model inference and parameter estimation

Model inference refers to the estimation of the parameters from given observations (see Section 3.2 for further details). For a full Bayesian estimation one must resort to calculating the normalization term in (3.2.1), which is usually intractable, even in the case of LDA, though it is a relatively simple model. To solve the problem we need approximate inference algorithms, such

as Laplace approximation [72], mean field variational expectation maximization [13] or Gibbs sampling [47, 94].

The authors of LDA applied variational expectation maximization for inference, later [47] proved that Gibbs sampling is faster with the same efficiency. In our proposed models, i.e., in multi-corpus LDA (see the details in Chapter 5) and in linked LDA (see details in Chapter 6) we also deal with Gibbs sampling, thus we treat it in more details.

Gibbs sampling

Gibbs sampling is an algorithm based on Markov chain Monte Carlo (MCMC) simulation. MCMC methods are able to simulate high-dimensional probability distributions by the stationary behaviour of a Markov chain [5]. During the process, one sample per transition is generated in the chain. The chain starts from an initial random state, then after a burn-in period it stabilizes by eliminating the influence of initialization parameters. The MCMC simulation of the probability distribution $p(\vec{x})$ is as follows: dimensions x_i are sampled alternately one at a time, conditioned on the values of all other dimensions denoted by $\vec{x}_{\neg i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$, that is:

1. choose dimension i (random or by cyclical permutation)
2. sample x_i from $p(x_i|\vec{x}_{\neg i})$

The requirement for building a Gibbs sampler is to know all the univariate conditionals (or full conditionals) $p(x_i|\vec{x}_{\neg i})$, which can be calculated as:

$$p(x_i|\vec{x}_{\neg i}) = \frac{p(\vec{x})}{\int p(\vec{x})\,\mathrm{d}x_i} \quad \text{with} \quad \vec{x} = \{x_i, \vec{x}_{\neg i}\}. \tag{3.4.5}$$

Gibbs sampler for LDA

In order to construct a Gibbs sampler for LDA, one has to estimate the probability distribution $p(\vec{z}|\vec{w})$ for $\vec{z} \in \mathcal{K}^N$, $\vec{w} \in \mathcal{V}^N$, where N denotes the set of word-positions in the corpus. This distribution is directly proportional to the joint distribution:

$$p(\vec{z}|\vec{w}) = \frac{p(\vec{z}, \vec{w})}{\sum_z p(\vec{z}, \vec{w})}. \tag{3.4.6}$$

The tricky part is the denominator, which is a summation over K^N terms. However, using Gibbs sampling, one requires only the full conditionals $p(z_i|\vec{z}_{\neg i}, \vec{w})$ in order to simulate $p(\vec{z}|\vec{w})$. First we derive the joint distribution:

$$p(\vec{w}, \vec{z}|\vec{\alpha}, \vec{\beta}) = p(\vec{w}|\vec{z}, \vec{\beta})p(\vec{z}|\vec{\alpha}), \tag{3.4.7}$$

because the first term is independent of $\vec{\alpha}$ due to the conditional independence of $\vec{w}$ and $\vec{\alpha}$ given $\vec{z}$, while the second term is independent of $\vec{\beta}$. The elements of the joint distribution can now be handled separately. The first term can be obtained from $p(\vec{w}|\vec{z}, \underline{\Phi})$, which is simply:

$$p(\vec{w}|\vec{z}, \underline{\Phi}) = \prod_{i=1}^{N} \varphi_{z_i, w_i} = \prod_{z=1}^{K} \prod_{t=1}^{V} \varphi_{z,t}^{N_{zt}}. \tag{3.4.8}$$

That is, the N words of the corpus are observed according to independent multinomial trials with parameters conditioned on the topic indices z_i. In the second equation we split the product over words into one product over topics and one over the vocabulary, separating the contributions of the topics. The term N_{zt} denotes the number of times that term t has been observed with topic z. The distribution $p(\vec{w}|\vec{z}, \vec{\beta})$ is obtained by integrating over $\underline{\Phi}$, which can be done component-wise using Dirichlet integrals within the product over z:

$$\begin{aligned} p(\vec{w}|\vec{z}, \vec{\beta}) &= \int p(\vec{w}|\vec{z}, \underline{\Phi}) p(\underline{\Phi}|\vec{\beta}) \, \mathrm{d}\underline{\Phi} \\ &= \int \prod_{z=1}^{K} \frac{1}{\Delta(\vec{\beta})} \prod_{t=1}^{V} \varphi_{z,t}^{N_{zt}+\beta_t - 1} \, \mathrm{d}\vec{\varphi}_z \\ &= \prod_{z=1}^{K} \frac{\Delta(\vec{N}_z + \vec{\beta})}{\Delta(\vec{\beta})}, \quad \vec{N}_z = \{N_{zt}\}_{t=1}^{V}. \end{aligned} \tag{3.4.9}$$

The topic distribution $p(\vec{z}|\vec{\alpha})$ can be derived from $p(\vec{z}|\underline{\Theta})$:

$$p(\vec{z}|\underline{\Theta}) = \prod_{i=1}^{N} \vartheta_{d_i, z_i} = \prod_{m=1}^{M} \prod_{z=1}^{K} \vartheta_{m,z}^{N_{mz}}, \tag{3.4.10}$$

where d_i refers to the document word w_i belongs to and N_{mz} refers to the number of times that topic z has been observed in document m. Integrating out $\underline{\Theta}$, we obtain:

$$\begin{aligned} p(\vec{z}|\vec{\alpha}) &= \int p(\vec{z}|\underline{\Theta}) p(\underline{\Theta}|\vec{\alpha}) \, \mathrm{d}\underline{\Theta} \\ &= \int \prod_{m=1}^{M} \frac{1}{\Delta(\vec{\alpha})} \prod_{z=1}^{K} \vartheta_{m,z}^{N_{mz}+\alpha_z - 1} \, \mathrm{d}\vec{\vartheta}_m \\ &= \prod_{m=1}^{M} \frac{\Delta(\vec{N}_m + \vec{\alpha})}{\Delta(\vec{\alpha})}, \quad \vec{N}_m = \{N_{mz}\}_{z=1}^{K}. \end{aligned} \tag{3.4.11}$$

The joint distribution therefore becomes:

$$p(\vec{w}, \vec{z}|\vec{\alpha}, \vec{\beta}) = \prod_{z=1}^{K} \frac{\Delta(\vec{N}_z + \vec{\beta})}{\Delta(\vec{\beta})} \cdot \prod_{m=1}^{M} \frac{\Delta(\vec{N}_m + \vec{\alpha})}{\Delta(\vec{\alpha})}. \tag{3.4.12}$$

Using (3.4.12) we can derive the update equation for the hidden variable:

$$p(z_i = k|\vec{z}_{\neg i}, \vec{w}) = \frac{p(\vec{z}, \vec{w})}{p(\vec{z}_{\neg i}, \vec{w})} = \frac{p(\vec{w}|\vec{z})}{p(\vec{w}|\vec{z}_{\neg i})} \cdot \frac{p(\vec{z})}{p(\vec{z}_{\neg i})} \tag{3.4.13}$$

$$\propto \frac{\Delta(\vec{n}_k + \vec{\beta})}{\Delta(\vec{n}_{z,\neg i} + \vec{\beta})} \cdot \frac{\Delta(\vec{n}_m + \vec{\alpha})}{\Delta(\vec{n}_{m,\neg i} + \vec{\alpha})} \tag{3.4.14}$$

$$= \frac{\frac{\Gamma(N_{kt}+\beta_t)}{\Gamma\left(\prod_{v=1}^{V}(N_{kv}+\beta_v)\right)} \cdot \frac{\Gamma(N_{mk}+\alpha_k)}{\Gamma\left(\prod_{z=1}^{K}(N_{mz}+\alpha_z)\right)}}{\frac{\Gamma(N_{kt}-1+\beta_t)}{\Gamma\left(\prod_{v=1}^{V}(N_{kv}+\beta_v)-1\right)} \cdot \frac{\Gamma(N_{mk}-1+\alpha_k)}{\Gamma\left(\prod_{z=1}^{K}(N_{mz}+\alpha_z)-1\right)}} \tag{3.4.15}$$

$$= \frac{N_{kt}^{\neg i} + \beta_t}{\sum_{v=1}^{V}(N_{kv} + \beta_v) - 1} \cdot \frac{N_{mk}^{\neg i} + \alpha_k}{\sum_{z=1}^{K}(N_{mz} + \alpha_z) - 1}, \tag{3.4.16}$$

where the superscript $N^{\neg i}$ denotes that the word or topic with index i is excluded from the corpus when computing the corresponding count. Note that only the terms of the products over m and k contain the index i; all the others cancel out. We used the $\Gamma(x) = (x-1)\Gamma(x-1)$ identity to get (3.4.16) from (3.4.15).

In any time, one can calculate the values of $\underline{\Theta}$ and $\underline{\Phi}$ from the given state of the Markov chain, i.e. from the current samples of $p(\vec{z}|\vec{w})$. This can be done as a posterior estimation, by predicting the distribution of a new topic-word pair $(\tilde{z} = k, \tilde{w} = t)$ that is observed in a document m, given state $(\vec{z}, \vec{w})$:

$$p(\tilde{z} = k, \tilde{w} = t|\vec{z}, \vec{w}) = p(\tilde{w} = t|\tilde{z} = k, \vec{z}, \vec{w}) \cdot p(\tilde{z} = k|\vec{z}, \vec{w}) \tag{3.4.17}$$

$$= \frac{p(\tilde{z} = k, \tilde{w} = t, \vec{z}, \vec{w})}{p(\vec{z}|\vec{w}))} = \frac{\frac{\Gamma(N_{kt}+1+\beta_t)}{\Gamma\left(\prod_{v=1}^{V}(N_{kv}+\beta_v)+1\right)} \cdot \frac{\Gamma(N_{mk}+1+\alpha_k)}{\Gamma\left(\prod_{z=1}^{K}(N_{mz}+\alpha_z)+1\right)}}{\frac{\Gamma(N_{kt}+\beta_t)}{\Gamma\left(\prod_{v=1}^{V}(N_{kv}+\beta_v)\right)} \cdot \frac{\Gamma(N_{mk}+\alpha_k)}{\Gamma\left(\prod_{z=1}^{K}(N_{mz}+\alpha_z)\right)}} \tag{3.4.18}$$

$$= \frac{N_{kt} + \beta_t}{\sum_{v=1}^{V}(N_{kv} + \beta_v)} \cdot \frac{N_{mk} + \alpha_k}{\sum_{z=1}^{K}(N_{mz} + \alpha_z)}. \tag{3.4.19}$$

Using the decomposition in (3.4.17), we can interpret its first factor as $\varphi_{k,t}$ and its second factor as $\vartheta_{m,k}$, hence:

$$\varphi_{k,t} = \frac{N_{kt} + \beta_t}{\sum_{v=1}^{V}(N_{kv} + \beta_v)}, \tag{3.4.20}$$

$$\vartheta_{m,k} = \frac{N_{mk} + \alpha_k}{\sum_{z=1}^{K}(N_{mz} + \alpha_z)}. \tag{3.4.21}$$

The Gibbs sampling in Algorithm 3.4.2 can be run using equations (3.4.16), (3.4.20) and (3.4.21). It uses three data structures, the matrix-like count tables N_{mz} and N_{zt}, which have dimensions $M \times K$ and $K \times V$ respectively and the sparse matrix-like topic assignment table $z_{m,n}$ with dimensions $M \times N_m$. The Gibbs sampling algorithm starts from a random topic-assignment state, then after a certain number of iterations (or alternatively the burn-in period) we can read the parameters of $\underline{\Theta}$ and $\underline{\Phi}$. Note that the determination of the required burn-in lengths is one of the drawbacks of MCMC approaches. There are several criteria to check that the Markov chain has converged (see [69]). We used cross entropy, which is a common evaluation measure in language modeling [46]. Normalized cross entropy (CE)[2] measures how close is the estimated model to the "true model", or in other terms, in what extent can the model be generalized to unseen data. More formally, given a test corpus $\tilde{\mathcal{D}}$ with M documents, then the CE with respect to a model build on a corpus $\mathcal{D}$ is:

$$\begin{aligned} CE(\mathcal{D}, \tilde{\mathcal{D}}) &= -\frac{1}{N}\log p(\tilde{\mathcal{D}}) = -\sum_{m=1}^{M}\frac{1}{N_m}\log p(\tilde{\vec{w}}_m) \\ &= -\sum_{m=1}^{M}\frac{1}{N_m}\sum_{n=1}^{N_m}\log\left(\prod_{k=1}^{K} p(w_n = t|z_n = k)p(z_n = k|d_m)\right) \\ &= -\frac{1}{N}\sum_{v=1}^{V} N_{mv}\log\left(\prod_{k=1}^{K}\varphi_{k,t}\vartheta_{m,k}\right). \end{aligned} \tag{3.4.22}$$

3.4.3 Unseen inference for LDA

Unseen inference refers to topic estimation of a collection of unseen documents $\tilde{\mathcal{D}}$ based on the previously built LDA model. For a document $\tilde{m} \in \tilde{\mathcal{D}}$, we need to estimate the posterior distribution of topics $\tilde{\vec{z}}$, where $\tilde{m} = \tilde{\vec{w}} \in V^{N_{\tilde{m}}}$ and $\tilde{\vec{z}} \in T^{N_{\tilde{m}}}$. Using the LDA model $L(\underline{\Phi}, \underline{\Theta})$,

$$p(\tilde{\vec{z}}|\tilde{\vec{w}}, L) = p(\tilde{\vec{z}}|\tilde{\vec{w}}, \vec{w}, \vec{z}), \tag{3.4.23}$$

[2] Another often cited measure is the perplexity, which is simply 2^{CE}.

Algorithm 3.4.2 Gibbs sampling algorithm for LDA from [56]

initialization: $N_{mz} \leftarrow N_{zt} \leftarrow 0$
for all documents $m \in [1, M]$ **do**
  for all words $n \in [1, N_m]$ in document m **do**
    sample topic index $z_{m,n} \sim Multinomial(1/K)$
    $N_{mz} \leftarrow N_{mz} + 1$
    $N_{zt} \leftarrow N_{zt} + 1$
{Gibbs sampling over burn-in period and sampling period}
while not finished **do**
  for all documents $m \in [1, M]$ **do**
    for all words $n \in [1, N_m]$ in document m **do**
      update counts for $w_{m,n} = t, z_{m,n} = z$:
      $N_{mz} \leftarrow N_{mz} - 1$
      $N_{zt} \leftarrow N_{zt} - 1$
      $\tilde{z} \sim p(z_i | \vec{z}_{\neg i}, \vec{w})$
      update counts for $w_{m,n} = t, z_{m,n} = \tilde{z}$:
      $N_{m\tilde{z}} \leftarrow N_{m\tilde{z}} - 1$
      $N_{\tilde{z}t} \leftarrow N_{\tilde{z}t} - 1$
    {check for convergence and compute parameters}
  if converged and L sampling iterations since last read out **then**
    compute $\underline{\Phi}$
    compute $\underline{\Theta}$
return

here $\vec{w} \in V^N, \vec{z} \in T^N$ are the word and topic vectors calculated on a training corpus $\mathcal{D}$. We first initialize Algorithm in 3.4.2 by randomly assigning topics to words in $\tilde{m}$, and initializing the count tables N_{mz}, N_{zt} with the counts from the model L. Then we perform a number of iterations with the Gibbs sampling update locally for the words i of $\tilde{m}$:

$$p\left(\tilde{\vec{z}}_i = k | \tilde{\vec{z}}_{\neg i}, \tilde{\vec{w}}, \vec{z}_{\neg i}, \vec{w}\right) = \frac{N_{kt} + \tilde{N}_{kt}^{\neg i} + \beta_t}{\sum_{v=1}^{V}(N_{kv} + \tilde{N}_{kv} + \beta_v)} \cdot \frac{N_{mk} + N_{\tilde{m}k}^{\neg i} + \alpha_k}{\sum_{z=1}^{K}(N_{mz} + N_{\tilde{m}z} + \alpha_z)}, \quad (3.4.24)$$

where the $\tilde{N}_{kt}$ table counts the observations of term t and topic k in the unseen document $\tilde{m}$. Applying equation (3.4.21) to the unknown document:

$$\vartheta_{\tilde{m},k} = \frac{N_{\tilde{m}k} + \alpha_k}{\sum_{z=1}^{K}(N_{\tilde{m}z} + \alpha_z)}. \quad (3.4.25)$$

This procedure is applicable to the whole unseen corpus $\tilde{\mathcal{D}}$, which is done by letting $\tilde{m}$ range over the $\tilde{\mathcal{D}}$.

Chapter 4

A Comparative Analysis of LSA and LDA

In this chapter we discuss the application of LSA and LDA in the Web page classification task. We report results on a comparison of these two approaches using different vocabularies consisting of links and text. Both models are evaluated using different numbers of latent topics. Finally, we evaluate a hybrid latent variable model that combines the latent topics resulting from both LSA and LDA. This new approach turns out to be superior to the basic LSA and LDA models. In our experiments with categories and pages obtained from the Open Directory Project (ODP)[1] web directory, the hybrid model achieves an averaged F value of 0.852 and an averaged AUC value of 0.96.

4.1 Related results

It has long been recognized that text and link features extracted from pages can help to discover Web communities, which often leads to the extraction of topically coherent subgraphs useful for clustering or classifying Web pages. Many algorithms based solely on link information have been proposed to partition hypertext documents [15, 54, 88], to identify and examine the structure of topics on the Web [23, 34, 45] and to categorize Web content [50]. Other algorithms use the hyperlink structure of the Web to find related pages [30, 77]. An approach, somewhat different from ours that combines latent variable models to identify topics on the Web is Link-PLSA-LDA [80].

LSA and LDA are based on different principles, see Subsection 2.2.2 and 3.4. On the one hand, LSA assumes that words and documents can be represented as points in the Euclidean space. On the other hand, LDA (like other statistical models) assumes that the semantic properties of words and documents are expressed in terms of probabilistic topics. Although some recent theoretical work has been carried out comparing Euclidean and probabilistic latent vari-

[1] http://www.dmoz.org

able models (e.g., [79]), this is the first attempt to the best of the authors' knowledge to provide a thorough empirical comparison of the two modeling approaches in the Web page classification task.

We applied LSA and LDA to uncover the hidden structure of a document collection. Documents can then be compared by means of their topics and therefore documents can have high similarity even if they do not share many common terms.

Contrary to the usual setup, we do not use links to propagate topics, but instead we treat them as words and build latent topic models on them. LDA with a vocabulary consisting of links was first considered in [124]. Their model, called SSN-LDA, is exactly our LDA model on links, applied in a social network context. Aside from this paper, we are not aware of any results on latent topic models built on links.

4.2 Experimental setup

Our goal is to compare LSA and LDA in a Web page classification task. For that purpose, we run a series of tests described next.

4.2.1 Input data

In order to run our comparison tests, we selected nearly 500 topics from ODP. The topics were selected from the third level of the ODP hierarchy. A number of constraints were imposed on this selection with the purpose of ensuring the quality of our test set. For each topic we collected all of its URLs as well as those in its subtopics. The total number of collected pages was more than 350K. After this selection, the topics were merged into 8 different categories, preserving only the main classification given by ODP: Arts, Business, Computers, Health, Science, Shopping, Society, Sports. We randomly split this collection of pages into training (80%) and test (20%) collections.

Text and links were extracted from the collected pages and used to generate two vocabulary sets. We refer to the vocabulary set based on text as $\mathcal{T}$ and to the vocabulary set based on links as $\mathcal{L}$. In order to generate $\mathcal{T}$, we extracted all terms from the selected Web pages and kept only the 30K terms that occurred with the highest frequency. This vocabulary was filtered further by eliminating stop-words[2] and keeping only terms consisting of alphanumeric characters, including those containing the hyphen, and the apostrophe. Documents with length less than 1000 terms were discarded. Finally, the text was stemmed[3]. The final number of words in $\mathcal{T}$ was 21308.

[2] http://www.lextek.com/manuals/onix/stopwords1.html.
[3] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/.

The vocabulary set $\mathcal{L}$ combines the incoming links and outgoing links associated with our corpus, that is the vocabulary consists of web pages linking to or linked by a page in our corpus. Incoming links were obtained using the Google Web API. In our classification tests, we took special care to filter out those links coming from topical directories. Finally, we extracted all the outgoing links from the pages and added them to $\mathcal{L}$. Links that occur in less than 10 documents were removed from our vocabulary and we only kept (train and test) documents with at least 3 links. The size of vocabulary $\mathcal{L}$ was 44561. It is important to notice that distinct portions of the training and test data are kept for the link and text vocabularies. By intersecting the two test sets (i.e. the text based and the link based) we obtained a common test set with 1218 pages.

4.2.2 Model construction

Using the training collection, we created models for LSA and LDA based on different vocabularies and using different number of latent topics. By taking this approach, we were able to compare the performance of our models across different dimensions. For LSA we used the $tf \times idf$ pivoted scheme described in (2.2.3) and applied SVD with dimensions 5, 15 and 30. LDA was run for 1000 iterations to generate 5, 15 and 30 latent topics.

In what follows we will use a simple convention to name the tested models. For example, an LSA model that uses text vocabulary and 15 latent topics is referred to as LSA-T-15, while one that uses link vocabulary for the same number of topics is named LSA-L-15. In addition, one could explore alternative text and link combinations, as well as the integration of latent topics from LSA and LDA. For example, by combining LDA-T-15 with LDA-L-30 we obtained the LDA-T-15-L-30 model, and by combining, LDA-L-15-T-30 with LSA-L-15-T-30 we obtained LDA-L-15-T-30-LSA-L-15-T-30.

In order to generate the LSA models we used the Lanczos code of `svdpack` [10] to run SVD. For LDA we used our home developed C++ code[4]. Once our models were generated, we used the Weka machine learning toolkit [119] with 10 fold cross validation to run two different binary classifiers: C4.5 and SVM, separately for every category. The *F* and *AUC* values presented in the rest of the chapter are averaged over the 8 categories.

4.3 Results

In this section we report the averaged *F* and *AUC* values for the tested models. Although we generated models with 5, 15 and 30 latent topics, we will omit the results for the models with 5 topics as they performed poorly. Tables 4.1 and 4.2 compare the performance of the LSA and LDA models using different text and link vocabulary combinations. We can observe that

[4] `http://www.ilab.sztaki.hu/ibiro/lda/`

for both models the text vocabulary is superior to the link vocabulary. This is not surprising considering that the number of links associated with each pages is usually much smaller than the number of terms. However, we observe that the models that combine text and link features are appreciably superior to those based only on text.

Experiments	SVM	C4.5
L-15	0.105/0.514	0.198/0.575
L-30	0.136/0.532	0.433/0.732
T-15	0.531/0.824	0.487/0.722
T-30	0.562/0.839	0.446/0.687
L-15-T-15	0.666/0.881	0.558/0.753
L-15-T-30	$\mathbf{0.710/0.894}$	0.561/0.755
L-30-T-15	0.671/0.882	0.594/0.783
L-30-T-30	0.708/0.893	0.579/0.768

Table 4.1: Averaged *F* and *AUC* values with LSA.

Experiments	SVM	C4.5
L-15	0.249/0.724	0.385/0.738
L-30	0.367/0.758	0.458/0.761
T-15	0.464/0.834	0.435/0.686
T-30	0.619/0.876	0.453/0.710
L-15-T-15	0.699/0.900	0.604/0.787
L-15-T-30	$\mathbf{0.765/0.938}$	0.571/0.756
L-30-T-15	0.687/0.896	0.594/0.771
L-30-T-30	0.757/0.921	0.575/0.767

Table 4.2: Averaged *F* and *AUC* values with LDA.

The most effective model for LSA is LSA-L-15-T-30, which combines 15 link based topics with 30 text based topics. Similarly, the best LDA model is LDA-L-15-T-30. Table 4.3 summarizes the improvements obtained when link features are included in the vocabulary with SVM classifier.

Method	*F*	*AUC*
LSA-T-30	0.562	0.839
LSA-L-15-T-30	0.710	0.894
Improvement	26.3%	6.6%
LDA-T-30	0.619	0.876
LDA-L-15-T-30	0.765	0.938
Improvement	23.6%	7.1%

Table 4.3: Comparison of text (T) and link (L) based classification results.

We also observe that LDA was superior to LSA for all the tested combinations. Table 4.4 shows the averaged *F* and *AUC* values for the best LSA and LDA configuration, using SVM.

Method	*F*	*AUC*
LSA-L-15-T-30	0.710	0.894
LDA-L-15-T-30	0.765	0.938
Improvement	7.7%	4.9%

Table 4.4: Averaged *F* and *AUC* values for the best LSA and LDA configuration.

Finally, we looked into the combination of the two latent variable models. Interestingly, by combining the best configurations of LDA and LSA we obtain the best model with an averaged *F* value of 0.852 and an averaged *AUC* value of 0.96. Table 4.5 summarizes these results.

The improvement registered by integrating both models points to the fact that the LDA and LSA models capture different aspects of the hidden structure of the corpus and that the combination of these models can be highly beneficial.

Method	*F*	*AUC*
LSA-L-15-T-30	0.710	0.894
LDA-L-15-T-30	0.765	0.938
LSA-L-15-T-30-LDA-L-15-T-30	**0.852**	**0.96**
Improvement LDA-LSA over LSA	20%	7.4%
Improvement LDA-LSA over LDA	11.3%	2.3%

Table 4.5: Averaged *F* and *AUC* values for the best LSA, LDA, and LSA-LDA combined configuration.

4.4 Conclusion

In our experiments we observe that although LDA is superior to LSA for all the tested configurations, the improvements achieved by combining latent structures from both approaches are noteworthy. Despite the different underlying assumptions of these two approaches and the apparent superiority of LDA, each one appears to have unique contribution for modeling text and links for classification.

Chapter 5

Multi-corpus LDA

In this chapter we introduce and evaluate a technique called Multi-corpus Latent Dirichlet Allocation (MLDA) for supervised semantic categorization of documents. In our setup, we assign topics separately for each category, and for a labeled training document only those topics are sampled that belong to its category. Thus, compared to the classical LDA that processes the entire corpus in one, we essentially build separate LDA models for each category with category-specific topics, and then these topic collections are put together to form a unified LDA model. For an unseen document the inferred topic distribution gives an estimation how much the document fits into the category.

We use this method for Web document classification. Our key results are 15% increase in *AUC* value in classification accuracy over $tf \times idf$ with SVM and 13% over the classical LDA baseline with SVM. Using a careful vocabulary selection method and heuristics to handle the effect that similar topics may arise in distinct categories, we achieve an improvement of 27% over $tf \times idf$ with SVM and 25% over LDA with SVM in *AUC*.

5.1 MLDA model

MLDA is a hierarchical method with two levels: categories and topics. Assume we have a supervised document categorization task with m categories. Every document is assigned to exactly one category, and this assignment is known only for the training corpus. For every category we assign a collection of topics of its own, and the union of these collections forms the topic collection of LDA. In LDA a Dirichlet parameter vector α is chosen for every document so that topics assigned to the document's words are drawn from a fixed multinomial distribution drawn from $\mathrm{Dir}(\alpha)$. In MLDA, we require for every training document that this Dirichlet parameter α has component zero for all topics outside the document's category, in order to achieve that only topics from the document's category are sampled to the document's words. This is equivalent to building separate LDA models for every category with category-specific topics. For an unseen

document d the fraction of topics in the topic distribution of d that belong to a given category measures how well d fits into that category. As a Dirichlet distribution allows only positive parameters, we will extend the notion of Dirichlet distribution in a natural way by allowing zeros. Although there exist hierarchical latent topic models [14, 111] similar to ours that tackle more than one layers, the advantage of MLDA is that it is built up from plain LDA's and no complicated hierarchical models need to be developed.

More formally, if $\gamma = (\gamma_1, \dots, \gamma_l, 0, \dots, 0) \in \mathbb{R}^n$ where $\gamma_i > 0$ for $1 \leq i \leq l$, then let the distribution $\mathrm{Dir}(\gamma)$ be concentrated on the subset $\{x \in \mathbb{R}^n : x_i = 0 \ \forall i > l, \sum_{1 \leq i \leq n} x_i = 1\}$, with distribution $\mathrm{Dir}(\gamma_1, \dots, \gamma_l)$. Thus for $p \sim \mathrm{Dir}(\gamma)$ we have that $p_i = 0$ for $i > l$ with probability 1, and $(p_1, \dots, p_l)$ is of distribution $\mathrm{Dir}(\gamma_1, \dots, \gamma_l)$. It can be checked in the deduction of [56] that the only property used in the calculus of Subsection 3.4.2 is that the Dirichlet distribution is conjugate to the multinomial distribution, which is kept for our extension, by construction. Indeed, if $x \sim \chi$ where $\chi \sim \mathrm{Dir}(\gamma_1, \dots, \gamma_l, 0, \dots, 0)$ with $\gamma_i > 0$ for $1 \leq i \leq l$, then for $i > l$ we have that $\chi_i = 0$ and thus $x_i = 0$ with probability 1. So the maximum a posteriori estimation of χ_i is

$$\frac{\gamma_i + x_i}{\sum_{1 \leq j \leq n} \gamma_j + x_j},$$

because the same holds for the classical case. To conclude, every calculation of the previous subsection still holds.

As $p(z_i = k | \vec{z}_{\neg i}, \vec{w}) = 0$ in Equation (3.4.16) if k has 0 Dirichlet prior, that is if it does not belong to the category of the document, the model inference procedure breaks down into making separate model inferences, one for every category. In other words, if we denote the collection of those training documents which were assigned category i by C_i, $1 \leq i \leq m$, then MLDA model inference essentially builds m separate LDA models, one for every C_i, with an appropriate choice of the topic number k_i. After all model inferences have been done, we have term distributions for all $k = \sum\{k_i : 1 \leq i \leq m\}$ topics.

Unseen inference is the same as for LDA. For an unseen document d, we perform Gibbs sampling and after a sufficient number of iterations, we calculate ϑ_d as in (3.4.21). For every category $1 \leq i \leq m$ we define

$$\xi_i = \sum_{j \in \mathcal{K}_i} \vartheta_{d,j}, \tag{5.1.1}$$

where $\mathcal{K}_i$ is the set of topic indices belonging to the category i. As ξ_i estimates how relevant category i is to document d, ξ_i is a classifier itself. We call this **direct classifier**, and measure its accuracy in terms of the *AUC* value, see Section 5.2. It is an appealing property of MLDA that right after unseen inference the resulting topic distribution directly gives rise to a classification. That is in contrast to, say, using plain LDA for categorization, where the topic distribution of the documents serve as features for a further advanced classifier.

MLDA also outperforms LDA in its running time. If there are k_i topics and N_i word positions in category i, then MLDA model inference runs in time $O(I \cdot \sum_{i=1}^{m} k_i N_i)$, where I is the number of iterations. On the contrary, LDA model inference runs in time $O(I \cdot kN)$ where $k = \sum_{i=1}^{m} k_i$ and $N = \sum_{i=1}^{m} N_i$. The more categories we have, the more is the gain. In addition, model inference in MLDA can be run in parallel. Unseen inferences in MLDA and LDA take about the same time, both are $O(IkN)$ where N denotes the total number of word positions in unseen documents.

5.1.1 ϑ-smoothing with Personalized PageRank

After model inference every document has an estimated topic distribution $\vec{\vartheta}$. Averaging these $\vec{\vartheta}$'s over the documents gives a distribution vector $\widehat{\vec{\vartheta}}$ that estimates the overall presence of the topics in the corpus. We call $\widehat{\vec{\vartheta}}$ the **observed importance distribution** of the topics in the corpus.

We expect that topics with a high importance value are somewhat "more important" than those with a small value, hence for a given category i we examine the top words of the best and worst three topics. Figure 5.1 shows an example of category "Science". The first three topics have the greatest importance score according to $\widehat{\vec{\vartheta}}$, while the last three have the smallest and for the former we can easily assign a meaningful label but not for the latter. The results are similar for the other main categories.

It is possible that MLDA infers two very similar topics in two distinct categories. In such a case it can happen that multi-corpus unseen inference for an unseen document puts very skewed weights on these two topics, endangering classification performance. To avoid such a situation, we apply a modified 1-step Personalized PageRank on the topic space, taking topic-similarity and the observed importance into account.

Our method for redistributing ϑ weights is based on a PageRank computation over a graph with edge weights defined by the Jensen-Shannon divergence of the topics. We define PageRank first and a Jensen-Shannon divergence similarity measure between two probability distributions afterwards.

PageRank [85] is a recursive algorithm that assigns numerical importance scores to Web pages. It was first used in the Google web search engine. The key concept is as follows: high PageRank value can be obtained if a page have lots of links from pages with high PageRank. PageRank is defined by the random surfer model. At a given time k, the random surfer is at page u and chooses a page v from the set of outgoing links of u uniformly at random. This determines his position at time $k+1$. The PageRank of a page u is the probability that the random surfer will arrive at that particular page. The former random walk defines a Markov chain on the Web graph, the states are the nodes (Web pages) from G and the stochastic transition matrix $\underline{P}$ is defined as $p_{u,v} = \frac{1}{d^+(u)}$, where $d^+(u)$ is the number of outlinks of page u. In order to ensure a

Science

Best topics			Worst topics		
"Physics"	"Archaeology"	"Math"			
theory	archaeology	graph	asin	@card@	nbsp
quantum	archaeological	matrix	var	e	pp
particle	ancient	zero	js	x	var
space	iron	equation	amz	o	mammillaria
prime	century	algorithm	nanotechnology	v	synonym
universe	@card@	representation	getelementbyid	cat	opuntia
physics	evidence	algebra	science	obj	ssp
object	museum	theorem	gift	endobj	mammal
equation	study	point	molecular	ref	parodia
system	excavation	finite	savedisowned	id	qzc
mass	cultural	theory	array	gene	menu
digit	archaeologist	integer	message	mpo	cactus
gravity	pyramid	algebraic	isowned	tra	componentart
x	rock	math	savedratings	sha	mu
energy	point	linear	innerhtml	jun	array
lattice	research	e	e	buie	mammalia
relativity	bc	define	customer	tabby	univ
dimension	valley	curve	registry	Jul	vitis
mechanic	survey	function	handlebuy	breed	qzk
sum	temple	elliptic	starmessages	null	pteris

Figure 5.1: Best and worst topics of category Science according to the observed importance distribution. Assigning a meaningful label to the first three columns is relatively easy, while the last three are quite fuzzy. @card@ is a replacement of numbers.

unique stationary distribution, $\underline{P}$ must be irreducible and aperiodic. The aperiodicity, i.e. the greatest common divisor of the length of all directed cycles in G is equal to 1, is always satisfied in practice. $\underline{P}$ is irreducible if and only if there is a directed path between any pair of pages in G. This condition can be fulfilled by adding a transition matrix $\underline{E} = \vec{e} \cdot \vec{r}^T$ to $\underline{P}$, where $\vec{e}$ is a vector whose elements are all equal to 1, and $\vec{r}$ is a vector whose elements are non-negative and sum to 1. The new transition matrix is:

$$\underline{A} = [c \cdot \underline{P} + (1 - c) \cdot \underline{E}]^T , \tag{5.1.2}$$

where $1 - c$ is the probability that the surfer jumps to a random page according to $\vec{r}$ rather than following an outgoing link. When the elements of $\vec{r}$ are equal we get the original PageRank (PR), while with a non-uniform $\vec{r}$ the result will be the personalized PageRank vector (PPR). The PageRank vector is the unique right eigenvector of $\underline{A}$ corresponding to eigenvalue 1.

The Jensen-Shannon divergence is a symmetric distance function between distributions with range $[0, 1]$ with the following definition:

Definition 7 (Jensen-Shannon divergence) *Given two discrete probability distribution*[1]*, p and q, the Jensen-Shannon divergence is defined as:*

$$JSD(p||q) = \frac{1}{2}D\left(p||m\right) + \frac{1}{2}D\left(q||m\right) \tag{5.1.3}$$

where $m = \frac{p+q}{2}$, and $D(p||q)$ is the Kullback-Leibler divergence.

Definition 8 (Kullback-Leibler divergence)

$$D(p||q) = \sum_{x \in \mathcal{X}} p(x) \log\left(\frac{p(x)}{q(x)}\right). \tag{5.1.4}$$

Next we define our ϑ-smoothing procedure. Let us define the topic graph as a complete graph over the topics as nodes where edges are weighted by the Jensen-Shannon divergence of the topics. Fix a document m. After unseen inference, its topic distribution is $\vec{\vartheta}_m$. We smooth $\vec{\vartheta}_m$ by distributing the components of $\vec{\vartheta}_m$ among themselves, as follows. For all topics $h \in \mathcal{K}$, we replace $\vartheta_{m,h}$ by

$$\vartheta_{m,h}^{new} = c \cdot \vartheta_{m,h} + \sum_{j \in \mathcal{K} \setminus h} S_j \cdot \frac{\widehat{\vartheta}_j}{JSD(\varphi_j, \varphi_h) + \epsilon} \cdot \vartheta_{m,j} \tag{5.1.5}$$

where c is a smoothing constant, $\widehat{\vec{\vartheta}}$ is the observed importance distribution, ε is a small constant to avoid dividing with zero, and φ_j, φ_h are the word distributions representing topics j and h, respectively. Note that $c = 1$ corresponds to no ϑ-smoothing. In order to make sure that $\vartheta_{m,j}^{new}$ is a distribution, a normalization term S_j is chosen to satisfy:

$$\sum_{h \in \mathcal{K} \setminus j} S_j \cdot \frac{\widehat{\vartheta}_j}{JSD(\varphi_j, \varphi_h) + \epsilon} \cdot \vartheta_{m,j} = (1 - c)\vartheta_{m,j}, \tag{5.1.6}$$

The experiments on ϑ-smoothing in Subsection 5.3.2 show slight improvement in accuracy if the vocabulary has small discriminating power among the categories. This is perhaps because it is more probable that similar topics are inferred in two categories if there are more words in the vocabulary with high occurrence in both. We mention that ϑ-smoothing can be applied to the classical LDA as well.

[1]Continuous case can be defined analogously.

5.2 Experimental setup

We have collected 290k documents from 8 categories of the DMOZ web directory[2]: Arts, Business, Computers, Health, Science, Shopping, Society, Sports. In our experiments, a document consists only of the text of the `HTML` page. After dropping documents with length less than 2000, we are left with 12k documents with total length of 64M.

For every category, we randomly split the collection of pages assigned to that category into training (80%) and test (20%) collections, and we denote the training corpus of the i^{th} category by C_i. We learn the MLDA model on the training corpus, that is, effectively, we build separate LDA models, one for every category. Then we carry out unseen inference on the test corpus. For every unseen document d we define two aggregations of the inferred topic distribution ϑ_d: we use ϑ_d itself as well as the category-wise ξ sum defined in (5.1.1). As we already noted, the category-wise sum ξ gives an estimation on the relevancy of category i for document d that we use as a direct classifier.

As baseline, we learn an LDA model with the same number of topics on the training corpus (without using the category labels), and then take the inferred ϑ values as feature on the test corpus.

We make experiments on how classifiers perform on the collection of these aggregated features. We classify every category separately by binary classifiers (linear SVM, C4.5 and BayesNet) using 10-fold cross validation to get the *AUC* value. We report the average *AUC* over the 8 categories. Classification is carried out for both ϑ and ξ as features.

Every run (MLDA model build, unseen inference and classification) is repeated 10 times to get variance of the *AUC* measure. These are at most 0.01 throughout, so we do not quote them individually.

The calculations were performed with the machine learning toolkit Weka [118] for classification and a home developed C++ code for LDA[3].

The computations were run on a Debian Linux machine with 20GB RAM and 1.8GHz Dual Core AMD Opteron 865 processor with 1MB cache.

5.2.1 Term selection

Although the importance of term selection in information retrieval and text mining has been proved crucial by several results, most papers on LDA-based language models do not put strong emphasis on the choice of the vocabulary. In this work we perform a careful term selection in order to find terms with high coverage and discriminability. There are several results published on term selection methods for text categorization tasks [42, 67]. However, we do not directly

[2] `http://www.dmoz.org/`
[3] `http://www.ilab.sztaki.hu/~ibiro/lda/`

apply these here, as our setup is different in that the features put into the classifier come from discovered latent topics, and are not derived directly from terms.

We keep only terms consisting of alphanumeric characters, hyphen, and apostrophe. We delete all stop-words enumerated in the Onix list[4]. We stem by TreeTagger [5].

Our feature selection procedure is as follows.

1. For every training corpus C_i we take the terms with the largest tf value (calculated within C_i). The resulting set of terms is denoted by W_i.

2. We unify these term collections over the categories, that is, let $W = \bigcup\{W_i : 1 \leq i \leq m\}$,

3. We drop those terms w from W for which the entropy of the normalized tf-vector over the categories exceeds a threshold H_{th}, that is, for which

$$\sum_{i=1}^{m} \frac{tf_i(w)}{tf(w)} \cdot \log\left(\frac{tf_i(w)}{tf(w)}\right) \geq H_{th}$$

   where $tf_i(w)$ is the term frequency of w calculated within C_i.

Term selection has two important aspects, discriminability and coverage. Note that step 1 takes care of the first and step 3 of the second.

We also made experiments with the unfiltered vocabulary (stop-wording and stemming the largest 30k terms in tf). For the thresholds set in our experiments, the size of the vocabulary and the average document length after filtering is shown in Table 5.1. The first line refers to the unfiltered case.

tf_{top}	H_{th}	$\lvert V\rvert$	$\widehat{d_l}$
none	none	21862	2602
15000	1.8	35996	1207
30000	1.8	89299	1329
30000	1.5	74828	637

Table 5.1: The size of the vocabulary $|V|$ and the average document length $\widehat{d_l}$ after filtering for different thresholds.

We mention that the running time of LDA is insensitive to the size of the vocabulary, so this selection is exclusively for enhancing performance.

[4] http://www.lextek.com/manuals/onix/stopwords1.html
[5] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

5.2.2 LDA inference

Model inference is performed separately for every training corpus C_i as described in Subsection 3.4.2, and then unseen inference is made for the test corpus w.r.t. the unified model. The number of topics k is chosen in three ways with the values given in Table 5.2:

1. $k_i = 50$ for all categories, referred as **const**
2. k_i is the number of subcategories of the category, referred as **sub**
3. k_i is the number of sub-sub-categories of the category, referred as **sub-sub**

category	**const**	**sub**	**sub-sub**
Arts	50	15	41
Business	50	23	71
Computers	50	17	44
Health	50	15	39
Science	50	12	58
Shopping	50	22	82
Society	50	20	58
Sports	50	14	28
sum, $k =$	400	138	421

Table 5.2: Three choices for topic number k.

The Dirichlet parameter β was chosen to be constant 0.1 throughout. For a training document in category i we chose α to be $50/k_i$ on topics belonging to category i and zero elsewhere. During unseen inference, α is defined accordingly, that is, for every topic z we have $\alpha_z = 50/k_i$ if z belongs to category i. The tests are run with and without ϑ-smoothing, with $c = 0.5, 0.75, 0.85$ and $\varepsilon = 0.001$. We apply Gibbs sampling for inference with 1000 iterations throughout.

5.3 Results

5.3.1 Plain MLDA vs LDA

As a justification for MLDA, we compared plain MLDA (no vocabulary filtering and ϑ-smoothing) with the classical LDA. The vocabulary is chosen to be unfiltered in both cases (see Subsection 5.2.1). For MLDA we tested all three variations for topic number (see Table 5.2). We show classification results with ξ features only since with ϑ the *AUC* values were about 5% worse. The classifiers were linear SVM, BayesNet and C4.5, as implemented in Weka, together with

the direct classification (defined in (5.1.1)). For LDA, the number of topics was $k = 138$, which is equal to the total number of topics in MLDA **sub**, and for a test document, the corresponding 138 topic probabilities served as features for the binary classifiers in the test corpus. Another baseline classifier is SVM over $tf \times idf$, run on the whole corpus with 10-fold cross validation. The *AUC* values are averaged over the 8 categories. The results are shown in Table 5.3.

	SVM	BayesNet	C4.5	direct
MLDA (const)	0.812	0.812	0.605	0.866
MLDA (sub)	0.820	0.826	0.635	**0.867**
MLDA (sub-sub)	0.803	0.816	0.639	0.866
LDA ($k = 138$)	0.765	0.791	0.640	–
SVM over $tf \times idf$	0.755	–	–	–

Table 5.3: Comparing plain MLDA with LDA in averaged *AUC* value.

The direct MLDA classifier outperforms the baselines and the classification methods on MLDA based ϑ features. Table 5.3 indicates that MLDA is quite robust to topic number. However, as topic number choice **sub** was the best, we used this one in later tests.

5.3.2 Vocabularies and ϑ-smoothing

We made experiments on MLDA to fine tune the vocabulary selection thresholds and to test performance of the ϑ-smoothing heuristic by a parameter sweep. Note that $S = 1$ in ϑ-smoothing corresponds to doing no ϑ-smoothing. We fixed 7 kinds of vocabularies (with different choices of tf_{top} and H_{th}) and the topic-number was chosen to be **sub** (see Table 5.2). We applied direct classification with results shown in Table 5.4.

vocabulary	$c = 1$	0.85	0.75	0.5
30000 / 1.8	0.954	0.955	0.956	**0.958**
30000 / 1.5	0.948	0.948	0.948	0.947
30000 / 1.0	0.937	0.937	0.936	0.934
15000 / 1.8	0.946	0.947	0.948	0.952
15000 / 1.2	0.937	0.937	0.937	0.936
10000 / 1.5	0.942	0.942	0.943	0.943
unfiltered	0.867	0.866	0.861	0.830

Table 5.4: Averaged *AUC* of different values c for ϑ-smoothing and the vocabulary parameters (tf_{top}/H_{th}).

It is apparent that our term selection methods result in large improvement in accuracy. Classification is more accurate if H_{th} and tf_{top} are larger. As both result in larger vocabularies, term selection should be conducted carefully to keep the size of the vocabulary big enough. Note that

the larger the entropy parameter H_{th} is, the more ϑ-smoothing improves performance. This is perhaps because of the fact that large H_{th} results in a vocabulary consisting of words with low discriminability among the categories, and thus topics in distinct categories may have similar word-distributions.

Every run (MLDA model build, unseen inference and classification) was repeated 10 times to get variance of the *AUC* measure. These were at most 0.01 throughout, so we decided not to quote them individually.

5.3.3 Running times

If the filtering parameters of the vocabulary are chosen to be $tf_{top} = 15000$ and $H_{th} = 1.8$, and the topic number is **sub** then model inference took 90 minutes for the biggest category Society (4.3M word positions), and 5 minutes for the smallest category Sports (0.3M word positions). Unseen inference took 339 minutes, with the same settings.

An example

To illustrate MLDA's performance, we show what categories MLDA inferred for the site `http://www.order-yours-now.com/`. As of July 2008, this site advertises a tool for creating music contracts, and it has ODP categorization Computers: Software: Industry-Specific: Entertainment Industry.

The 8 category-wise ξ features of MLDA defined in (5.1.1) that measure the relevance of the categories are given in Table 5.5. We feel that MLDA's categorization is on a par with or perhaps better than that of ODP. The top ODP category is Computers, perhaps because the product is sold as a computer program. On the contrary, MLDA suggests that the site mostly belongs to Arts and Shopping, which we feel appropriate as it offers service for musicians for profit. MLDA also detects the Business concept of the site, however, Shopping is given more relevance than Business because of the informal style of the site.

The parameters for MLDA were set as follows: $tf_{top} = 15000$, $H_{th} = 1.8$, $c = 0.5$ for ϑ-smoothing, **sub** as topic number.

5.4 Conclusion and future work

In this chapter we have described Multi-corpus LDA (MLDA), a way to apply LDA for supervised text categorization by viewing it as a hierarchical topic model. Essentially, separate LDA models are built for each category with category-specific topics, then these models are unified, and inference is made for an unseen document w.r.t. this unified model. As a key observation,

category	$\sum \vartheta$
Arts	0.246
Shopping	0.208
Business	0.107
Health	0.103
Society	0.096
Computers	0.094
Sports	0.076
Science	0.071

Table 5.5: Relevance of categories for site `http://www.order-yours-now.com/`, found by MLDA.

the topic unification method significantly boosted performance by avoiding overfitting to a large number of topics, requiring lower running times.

Chapter 6

Linked LDA and sparse Gibbs samplers

In this chapter we demonstrate again the applicability of LDA for classifying large Web document collections. One of the main results is a novel influence model, called linked LDA that gives a fully generative model of the document content taking linkage into account. In this setup, topics propagate along links in such a way that linked documents directly influence the words in the linking document. As another main contribution, we develop LDA specific boosting of Gibbs samplers resulting in a significant speedup in our experiments. The inferred LDA model can be applied for classification as dimensionality reduction similarly to LSA. In addition, the model yields link weights that can be applied in algorithms to process the Web graph; as an example we deploy LDA link weights in stacked graphical learning. By using BayesNet classifier, in terms of the AUC of classification, we achieve 4% improvement over the classical LDA with BayesNet and 18% over $tf \times idf$ with SVM. Our Gibbs sampling strategies yield about 5-10 times speedup with less than 1% decrease in accuracy in terms of likelihood and AUC of classification.

Recently several models extend LDA to exploit links between web documents or scientific papers [25, 33, 37, 81]. In these models the term and topic distributions may be modified along the links. All these models have the drawback that every document is thought of either citing or cited, in other words, the citation graph is bipartite, and influence flows only from cited documents to citing ones.

In this research we develop the **linked LDA** model, in which each document can cite to and be cited by others and thus be influenced and influence other documents. Linked LDA is very similar to the citation influence model of Dietz, Bickel and Scheffer [33] with the main difference that in our case the citation graph is not restricted to be bipartite. This fact and its consequences are the main advantages of linked LDA, namely that the citation graph is homogeneous, one does not have to take two copies, citing and cited, of every document, and finally that influence may flow along paths of length more than one, a fact that gives power to learning over graphs [64, 125]. In addition, we give a flexible model of all possible effects,

including cross-topic relations and link selection. As an example, we may model the fact that topics Business and Computer are closer to one another than to Health as well as the distinction between topically related and unrelated links such as links to software to view the content over a Health site. The model may also distinguish between sites with strong, weak or even no influence from its neighbors. The linked LDA model is described in full detail in Subsection 6.2.

We demonstrate the applicability of linked LDA for text categorization, an application which is explicitly mentioned in [13] but, to the author's knowledge, justified prior to our work only in special applications [12]. The inferred topic distributions of documents are used as features to classify the documents into categories. In the linked LDA model, a weight is inferred for every link. In order to validate the applicability of these edge weights, we show that their usage improves the performance of stacked graphical classification, a meta-learning scheme introduced in [64] (for a brief overview we refer to Subsection 2.3.4).

The crux in the scalability of LDA for large corpora lies in the understanding of the Gibbs sampler for inference. In the first application of the Gibbs sampler to LDA [47] as well as in the fast Gibbs sampler [91] the unit of sampling or, in other terms, a transition step of the underlying Markov chain, is the redrawing of one sample for a single term occurrence. The storage space and update time of all these counters prohibit sampling for very large corpora. Since, however, the order of sampling is neutral, we may group occurrences of the same term in one document together. Our main idea is then to re-sample each of these term positions in one step and, to assign a joint storage to them. We introduce three strategies: for **aggregated** sampling we store a sample for each position as above but update all of them in one step for a word, for **limit** sampling we update a topic distribution for each distinct word instead of drawing a sample, while for **sparse** sampling we randomly skip some of these words. All of these methods result in a significant speedup of about 5-10 times, with less than 1% decrease in accuracy in terms of likelihood and *AUC* of classification. The largest corpus where we could successfully perform classification using these boostings consisted of 100k documents (that is, web sites with a total of 12M pages), and altogether 1.8G term positions.

To assess the prediction power of the proposed features, we run experiments on a host-level aggregation of the `.uk` domain, which is publicly available through the Web Spam Challenge [21]. We perform topical classification into one of 11 top-level categories of the ODP. Our techniques are evaluated along several alternatives and, in terms of the *AUC* measure, yield an improvement of 4% over the classical LDA and 18% over $tf \times idf$ with SVM (here BayesNet is used on linked LDA based features).

6.1 Related results

We compare our linked LDA model to preexisting extensions of PLSA and LDA that jointly model text and linkage as well as the influence of topics along links. The first such model is PHITS defined by Cohn and Hoffman [25]. The mixed membership model of Erosheva, Fienberg and Lafferty [37] can be thought of as an LDA based version of PHITS. Common to these models is the idea to infer similar topics to documents that are jointly similar in their "bag of words" and link adjacency vectors. Later, several similar link based LDA models were introduced, including the copycat model, the citation influence model by Dietz, Bickel and Scheffer [33] and the link-PLSA-LDA and pairwise-link-LDA models by Nallapati, Ahmed, Xing and Cohen [81]. These two results extend LDA over a bipartition of the corpus into citing and cited documents so that influence flows along links from cited to citing documents. They are shown to outperform earlier methods [33, 81]. While these models generate topical relation for hyperlinked documents, in a homogeneous corpus one has to duplicate each document and infer two models for them. This is in contrast to the linked LDA model that treats citing and cited documents identically. As a completely different direction for link-based LDA models, we mention the results [109, 123, 124] which give a generative model for the links of a network, with no words at the nodes.

We also compare the performance of our results to general classifiers aided by Web hyperlinks. Relational learning methods (presented, for instance, in [44]) also consider existing relationships between data instances. The first relational learning method designed for topical web classification was proposed by Chakrabarti, Dom and Indyk [22] and improved by Angelova and Weikum [6]. Several subsequent results [95, and the references therein] confirm that classification performance can be significantly improved by taking into account the labels assigned to neighboring nodes. In our baseline experiments we use the most accurate hypertext classifiers [20] obtained by stacked graphical learning, see Subsection 2.3.4 for details. Performance of stacked graphical learning is evaluated in Subsection 6.5.3 both with various graph based edge weights [28] and with those inferred by our linked LDA model.

Another main contribution of this work is three new efficient inference methods. Upon introducing LDA, Blei, Ng, Jordan [13] proposed a variational algorithm for inference. Later, several other methods were described for inference in LDA, namely collapsed Gibbs sampling [47], expectation propagation, and collapsed variational inference [112]. Besides, [82] suggests methods on how Gibbs sampling can be applied in a paralleled environment.

Among the above collapsed Gibbs sampling methods, fastest convergence is achieved by that of Griffiths and Steyvers [47]. To the author's knowledge, prior to our result there has been one type of attempt to speed up LDA inference in general, and LDA based Gibbs sampling in particular. Porteous, Newman, Ihler, Asuncion, Smyth, Welling [91] modify Gibbs sampling such that it gives the same distribution by using search data structure for sample updates. They

show significant speedup for large topic numbers.

6.2 Linked LDA model

Next we extend LDA to model the effect of a hyperlink between two documents on topic and term distributions. The key idea is to modify the topic distribution of a position on the word plate based on a link from the current document on the document plate. For each word position, we select either an outlink or the document itself to modify the topic distribution of the original LDA model.

Formally we introduce linked LDA over the notations of Subsection 3.4. Links are represented by a directed graph with inlinks for cited and outlinks for citing documents. Our model also relies on the LDA distributions $\vec{\varphi}_k$ and $\vec{\vartheta}_m$. We introduce an additional distribution $\vec{\chi}_m$ on the set $\mathcal{S}_m$, which contains the document m and its outneighbors. $\vec{\chi}_m$ is sampled from $\mathrm{Dir}(\vec{\gamma}_m)$, where $\vec{\gamma}_m$ is a positive smoothing vector of size $|\mathcal{S}_m|$. The complete generative model can be seen in 6.2.1 and the corresponding BayesNet is in Figure 6.1.

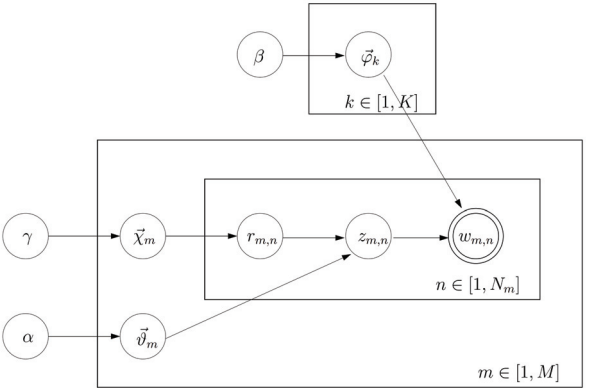

Figure 6.1: Linked LDA as a Bayesian Network.

Note that for sake of a unified treatment, m itself can be an influencing document of itself. This is in contrast to the citation influence model of [33], where for every word a Bernoulli draw decides whether the influencing document is m itself or an outneighbor of it.

We describe the Gibbs sampling inference procedure for linked LDA. The goal is to estimate the distribution $p(\vec{r}, \vec{z}|\vec{w})$ for $\vec{r} \in \mathcal{D}^N, \vec{z} \in \mathcal{T}^N, \vec{w} \in \mathcal{V}^N$ where $N = \sum_{i=1}^{M} N_i$ denotes the set

Algorithm 6.2.1 Generative model for linked LDA

"topic plate"
for all topics $k \in [1, K]$ **do**
  sample mixture components $\vec{\varphi}_k \sim Dir(\vec{\beta})$
"document plate"
for all documents $m \in [1, M]$ **do**
  sample mixture proportion $\vec{\chi}_m \sim Dir(\vec{\gamma})$
  sample mixture proportion $\vec{\vartheta}_m \sim Dir(\vec{\alpha})$
  "word plate"
  for all words $n \in [1, N_m]$ in document m **do**
    sample influencing document index $r_{m,n} \sim Multinomial(\vec{\chi}_m)$
    sample topic index $z_{m,n} \sim Multinomial(\vec{\vartheta}_{r_{m,n}})$
    sample term for word $w_{m,n} \sim Multinomial(\vec{\varphi}_{z_{m,n}})$
return

of word positions in the documents. In Gibbs sampling one has to calculate $p(z_{m,n} = k, r_{m,n} = l|\vec{z}_{\neg(m,n)}, \vec{r}_{\neg(m,n)}, \vec{w})$ for $m \in [1, M]$ and $n \in [1, N_m]$. First, it can be shown that

$$p(\vec{r}, \vec{z}, \vec{w}|\vec{\alpha}, \vec{\beta}, \vec{\gamma}) = \prod_{m=1}^{M} \frac{\Delta(\vec{N}_m^r + \vec{\gamma}_m)}{\Delta(\vec{\gamma}_m)} \cdot \prod_{m=1}^{M} \frac{\Delta(\vec{N}_d^z + \vec{\alpha})}{\Delta(\vec{\alpha})} \cdot \prod_{z=1}^{K} \frac{\Delta(\vec{N}_z^w + \vec{\beta})}{\Delta(\vec{\beta})}. \tag{6.2.1}$$

Here $\vec{N}_m^r$ is the count vector of influencing documents appearing at the words of document m, $\vec{N}_d^z$ is the count vector of observed topics influenced by m, and $\vec{N}_z^w$ is the count vector of words with topic z.

By division, (6.2.1) gives an iteration in the Gibbs sampling as follows:

$$p(z_{m,n} = k, r_{m,n} = l|\vec{z}_{\neg(m,n)}, \vec{r}_{\neg(m,n)}, \vec{w}) = \frac{p(\vec{z}, \vec{r}, \vec{w})}{p(\vec{z}_{\neg(m,n)}, \vec{r}_{\neg(m,n)}, \vec{w}_{\neg(m,n)})} \cdot$$

$$\frac{1}{p(\vec{w}_{m,n}|\vec{z}_{\neg(m,n)}, \vec{r}_{\neg(m,n)}, \vec{w}_{\neg(m,n)})},$$

thus

$$p(z_{m,n} = k, r_{m,n} = l|\vec{z}_{\neg(m,n)}, \vec{r}_{\neg(m,n)}, \vec{w}) \propto \frac{p(\vec{z}, \vec{r}, \vec{w})}{p(\vec{z}_{\neg(m,n)}, \vec{r}_{\neg(m,n)}, \vec{w}_{\neg(m,n)})} =$$

$$\frac{N_{lk}^{\neg(m,n)} + \alpha_k}{\sum_{z=1}^{K}(N_{lz} + \alpha_z) - 1} \cdot \frac{N_{ml}^{\neg(m,n)} + \gamma_l}{\sum_{r=1}^{S_m}(N_{mr} + \gamma_r) - 1} \cdot \frac{N_{kt}^{\neg(m,n)} + \beta_t}{\sum_{v=1}^{V}(N_{kv} + \beta_v) - 1}, \tag{6.2.2}$$

where N_{ml} denotes the number of words in document m influenced by outneighbor $l \in S_m$, similarly, N_{lk} is the number of words in document l with topic k, and N_{kt} is the count of word

t in topic k in the corpus. The superscript $^{\neg(m,n)}$ means that the index (m,n) is excluded from the counting.

Note that, this "two-coordinate" sampling is against the general Gibbs procedure that resamples one coordinate at a time, however, we experienced lower running times and no deterioration in inference quality. For the sake of completeness we specify the straightforward one-coordinate sampling, i.e. sample on $z_{m,n} = k$ and $r_{m,n} = l$ one after another:

$$p(z_{m,n} = k | \vec{z}_{\neg(m,n)}, \vec{r}, \vec{w}) \propto \frac{N_{lk}^{\neg(m,n)} + \alpha_k}{\sum_{z=1}^{K}(N_{lz} + \alpha_z) - 1} \cdot \frac{N_{kt}^{\neg(m,n)} + \beta_t}{\sum_{v=1}^{V}(N_{kv} + \beta_v) - 1}, \tag{6.2.3}$$

and

$$p(r_{m,n} = l | \vec{z}, \vec{r}_{\neg(m,n)}, \vec{w}) \propto \frac{N_{lk}^{\neg(m,n)} + \alpha_k}{\sum_{z=1}^{K}(N_{lz} + \alpha_z) - 1} \cdot \frac{N_{ml}^{\neg(m,n)} + \gamma_l}{\sum_{r=1}^{S_m}(N_{mr} + \gamma_r) - 1}. \tag{6.2.4}$$

Similarly to LDA, we stop after a sufficient number of iterations with the current topic assignment sample $\vec{z}$, and estimate φ as in (3.4.20), ϑ as in (3.4.21), and χ by

$$\chi_{ml} = \frac{N_{ml} + \gamma_{ml}}{\sum_{r=1}^{|S_m|}(N_{mr} + \gamma_{mr})}. \tag{6.2.5}$$

For an unseen document $\tilde{m}$, distributions ϑ and χ can be estimated exactly as in (3.4.21) and (6.2.5) once we have a sample from its word topic assignment $\vec{z}$. Sampling $\vec{z}$ can be performed with a similar method as above, but now only for the positions i in $\tilde{m}$.

6.3 Fast Gibbs sampling heuristics

In this section we describe three strategies for a faster inference for both the original and the linked LDA models. The methods modify the original Gibbs sampling procedure [47]. For simplicity, we describe them for the classical LDA setting. The speedup obtained by these boostings is evaluated in Subsection 6.5.1.

All of our methods start by sorting the words of the original documents so that sampling is performed subsequently for the occurrences of the same word. We introduce additional heuristics to compute the new samples for all occurrences of the same word in a document at once.

In **aggregated** Gibbs sampling we calculate the conditional topic distribution F as in Equation (3.4.16) for the first occurrence i of a word in a given document. Next we draw a topic

from F for *every* position with the same word without recalculating F, and update all counts corresponding to the same word. In this way, the number of calculations of the conditional topic distributions is the number of different terms in the document instead of the length of the document, moreover, the space requirement remains unchanged. This can be further improved by maintaining the aggregated topic count vector for terms with large frequency in the document, instead of storing the topic at each word.

Limit Gibbs sampling is based on the "bag of words" model assumption suggesting that the topic of a document remains unchanged by multiplying all term frequencies by a constant. In the limit hence we may maintain the calculated conditional topic distribution F for the set of all occurrences of a word, without drawing a topic for every single occurrence. Equation (3.4.16) can be adapted to this setting by a straightforward redefinition of the N counts. Similarly to aggregated Gibbs sampling, limit sampling may result in compressed space usage depending on the size and term frequency distribution of the documents.

It is easy to check that if the topic distributions for all positions are uniform then we get an instable fixed point of limit Gibbs sampling, provided both α and β are constant. Clearly, with large probability, these fixed points can be avoided by selecting biased initial topic distributions. We never encountered such instable fixed points during our experiments.

Sparse sampling with sparsity parameter ℓ is a lazy version of limit Gibbs sampling where we ignore some of the less frequent terms to achieve faster convergence on the more important ones. On every document m with d_m words, we sample d_m/ℓ times from a multinomial distribution on the distinct terms with replacement by selecting a term by a probability proportional to its term frequency tf_w in the document. Hence with $\ell = 1$ we expect a performance similar to limit Gibbs sampling, while large ℓ results in a speedup of about a factor of ℓ with a trade-off of lower accuracy.

Aggregated Gibbs sampling has the required distribution $p(\vec{z}|\vec{w})$ as its unique stationary distribution. Indeed, it is a random walk over a finite state Markov chain which is irreducible and aperiodic as $\alpha, \beta > 0$, implying that it has a unique stationary distribution, which is necessarily the distribution $p(\vec{z}|\vec{w})$ by construction.

As for limit Gibbs sampling, we rely on the assumption, that in the bag of words model, multiplying all term frequencies in the corpus by a constant the semantic meaning of the documents change only moderately. As limit Gibbs sampling arises as a limit of aggregated Gibbs sampling by tending this constant to infinity, its stationary distribution is very close to what the aggregated version samples, that is, the required $p(\vec{z}|\vec{w})$. This argument is justified by our measurements in Section 6.5.

Laziness clearly keeps the above arguments valid, thus aggregated sparse Gibbs sampling has stationary distribution $p(\vec{z}|\vec{w})$, and sparse sampling the same as limit.

6.4 Experimental setup

6.4.1 The data set and classifiers

In our experiments we use the 114k node host-level aggregation of the WEBSPAM-UK2007 `.uk` domain crawl, which consists of 12.5M pages and is publicly available through the Web Spam Challenge [21]. We perform topical classification into one of the 14 top-level English language categories of the Open Directory Project (ODP) while excluding category "World" containing non-English language documents. If a site contains a page registered in ODP with some top category, then we label it with that category. In case of a conflict we choose a random page of the site registered in ODP and label its site with its top category. In this way we could derive category label for 32k documents.

We perform the usual data cleansing steps prior to classification. After stemming by TreeTagger[1] and removing stop words by the Onix list[2], we aggregate the words appearing in all HTML pages of the sites to form one document per site. We discard rare terms and keep the 100k most frequent ones to form our vocabulary. We discard all hosts that become empty that is those consisting solely of probably only a few rare terms. Finally we weight directed links between hosts by their multiplicity. For every site we keep only at most 50 outlinks with largest weight.

We call this the **big corpus**, as it contains 1.8G positions, far more than the usual corpora on which LDA experiments have been carried out. Since the big corpus is infeasible for the baseline experiments, we also form a **small corpus** consisting of the labeled hosts only, to be able to compare our results with the baseline. We use the most frequent 20k terms as vocabulary and keep only the 10 largest weight outlink for each host. Note that even the small corpus is large enough to cause efficiency problems for the baseline classifiers.

We chose 11 out of the 14 categories to apply classification to them, as the other 3 categories were very small in our corpus. In our experiments we perform binary classification for all of these 11 big categories. We use the linear kernel SVM, the C4.5 decision tree and the BayesNet implementation of the machine learning toolkit Weka [118]. We use 10-fold cross validation and measure performance by the average *AUC* over the 11 classes. Every run ((linked) LDA model build and classification) is repeated 10 times to get variance of the *AUC* classification performance.

[1] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`
[2] `http://www.lextek.com/manuals/onix/stopwords1.html`

6.4.2 Baseline classifiers

As the simplest baseline we use the $tf \times idf$ vectors with SVM for the small corpus, as it took a prohibitively long time to run it on the big corpus. Another baseline is to use the LDA delivered ϑ topic distributions as features with the classifier BayesNet.

Recently a large number of relational learning methods were invented, however, their comparison is beyond the scope of this thesis. Instead we concentrate on the stacked graphical learning method [64] that reaches best performance for classifying Web spam over this corpus [20]. We use **cocitation** to measure node similarities. We may use both the input directed graph and its transpose by changing the direction of each link. We will refer to these variants as **directed** and **reversed** versions.

6.4.3 LDA inference

In our LDA inference, the following parameter settings are used. The number of topics is chosen to be $k = 30$. The Dirichlet parameter vector components $\vec{\beta}_j$ are constant $200/|V|$, and $\vec{\alpha}$ is also constant with values $50/k$. For linked LDA we also consider directed links between the documents. For a document d, the smoothing parameter vector γ_d is chosen in such a way that

$$\gamma_d(c) \propto w(d \to c) \quad \text{for all } c \in S_d, c \neq d \text{ and}$$

$$\gamma_d(d) \propto 1 + \sum_{c \in S_d, c \neq d} w(d \to c)$$

, so that $\sum_{c \in S_d} \gamma_d(c) = |d|/p$, where $|d|$ is the number of word positions in d (the document length), and $w(d \to c)$ denotes the multiplicity of the $d \to c$ link in the corpus. We chose $p = 10$ to get moderately high Dirichlet smoothing parameters.

We take the ϑ topic distribution vectors as features for the documents and use the classifiers of Subsection 6.4.1. For Gibbs sampling in LDA and linked LDA we apply the baseline as well as the aggregated, limit and sparse heuristics. Altogether this results in eight different classes of experiments.

As another independent experiment, we also measure the quality of the inferred edge weight function χ. This weight can be used in the stacked graphical classification procedure of Subsection 6.4.2 over the ϑ topic distribution feature and the $tf \times idf$ baseline classifiers, for both the link graph and its reversed version.

The same experiments are performed with the link-PLSA-LDA model [81], using the C-code kindly provided to us by Ramesh Nallapati. We also compared the running time of our Gibbs sampling strategies with the fast collapsed Gibbs sampling method of [91], using the

C-code referred to therein[3]. For results, see Subsection 6.5.1.

We developed an own C++-code for LDA and linked LDA containing plain Gibbs sampling and the three Gibbs sampling boostings proposed in this research. This code is publicly available[4] together with the used ODP labels of the `.uk` sites. The computations were run on a Debian Linux machine with 50GB RAM and 1.8GHz Dual Core AMD Opteron 865 processor with 1MB cache.

6.5 Results

6.5.1 Speedup with the Gibbs sampling strategies

Applying aggregated, limit and sparse Gibbs samplings results in an astonishing speedup, see Tables 6.1-6.2. Experiments were carried out on the small corpus, and the models were run with $k = 30$ topics.

Gibbs sampler	LDA	linked LDA
plain	1000	1303
aggregated	193	1006
limit	190	970
sparse ($\ell = 2$)	135	402
sparse ($\ell = 5$)	105	241
sparse ($\ell = 10$)	91	171
sparse ($\ell = 20$)	84	129
sparse ($\ell = 50$)	80	107

Table 6.1: Average CPU times for one iteration of the Gibbs sampler in seconds.

model / sampler	time
LDA / fast compressed Gibbs [91]	949
link-PLSA-LDA / var. inf. [81]	19,826

Table 6.2: Average CPU times for one iteration for baseline models and samplers in seconds.

Observe that the speedup of aggregated Gibbs sampling is striking, and this does not come at the expense of lower accuracy. Indeed, results of Subsections 6.5.2 and 6.5.3 show less than 1% decrease in terms of likelihood and *AUC* value after 50 iterations when using aggregated Gibbs sampling. The advantage of limit sampling over aggregated sampling (no need to draw from the distribution) turns out to be negligible. For sparse Gibbs sampling the speedup is a

[3] `http://www.ics.uci.edu/~iporteou/fastlda/`
[4] `http://www.ilab.sztaki.hu/~ibiro/linkedLDA/`

straightforward consequence of the fact that we skip many terms during sampling, and thus the running time is approximately a linear function of $1/\ell$, where ℓ is the sparsity parameter. The constant term in this linear function is apparently approximately 75 seconds for LDA and linked LDA, certainly, this is the time needed to iterate through the 16G memory. As for the choice $\ell = 10$, note that the running time of one iteration for LDA is only 9% of the one with plain Gibbs sampling, and still, the accuracy measured in *AUC* is only 2% worse as seen in Table 6.4.

We point out that though we used the same C-code as [91], our measurement on fast Gibbs sampling demonstrates a somewhat poorer performance than the results presented in [91], even if one takes into account that fast Gibbs sampling is proved to have better performance with a large number of topics ($k \approx 1000$) (with $k = 100$ topics we experienced 2100 seconds on average). On the other hand, fast Gibbs sampler gets faster and faster for the consecutive iterations: in our experiments with $k = 30$ topics we measured 2656 seconds for the first iteration (much worse than for plain Gibbs) and 757 seconds for the 50^{th}. Thus the calculated average would be better with more iterations – for which, however, there is no real need according to the observations in Subsection 6.5.2.

As the number of iterations with variational inference is usually chosen to be around 50, the same as for Gibbs sampling, we feel the above running times for Gibbs sampling and link-PLSA-LDA with its variational inference are comparable.

6.5.2 Likelihood and convergence of LDA inference

Figures 6.2-6.4 show the convergence of the likelihood and the *AUC* for BayesNet, for some combinations of LDA and linked LDA models run with various Gibbs samplers. The plots range over 50 iterations, and we have stopped inference after every 2 iterations and calculated the *AUC* of a BayesNet classification over the ϑ features, and the likelihood (as described in Subsections 3.4.2 and 6.2). The number of topics was $k = 30$ for all models, and the parameters were as described in Subsection 6.4.3.

The high pairwise correlation of the otherwise quite different accuracy measures, likelihood and *AUC* values in Figure 6.2 are very interesting. This is certainly due to the fact that after the topic assignment stabilizes, there is only negligible variance in the ϑ features. This behavior indicates that the widely accepted method of stopping LDA iterations right after the likelihood has stabilized can be used even if the inferred variables (ϑ in our case) are later input to other classification methods.

The usual choice for the number of Gibbs sampling iterations for LDA is 500-1000. Thus it is worth emphasizing that in our experiments, both likelihood and accuracy stabilizes after only 20-30 iterations. This is in accordance with the similar experiments of [47, 115], which found that for classical LDA, likelihood stabilizes after 50-100 iterations, over various corpora. As a consequence, we chose 50 as the number of iterations for the next experiments.

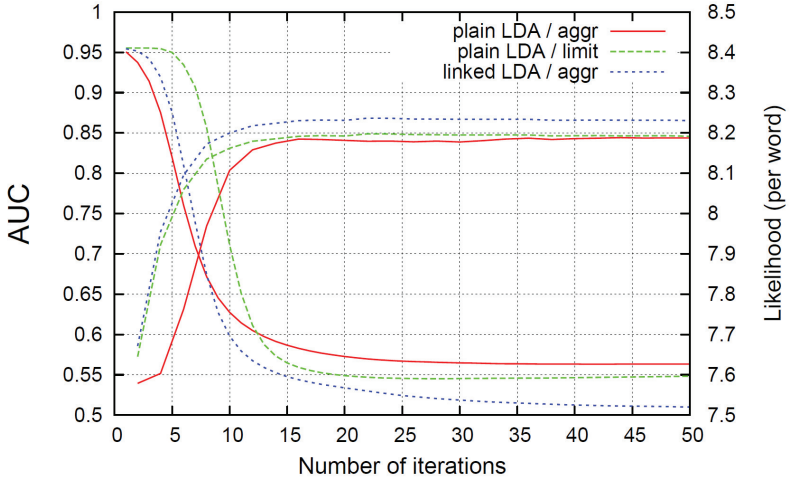

Figure 6.2: Correlation of the likelihood (the lower the better) and the *AUC* for BayesNet (the higher the better) for three choices of model and sampler combinations.

Figure 6.3 demonstrates that the linked LDA model with plain, aggregated and limit samplers overperform classical LDA by about 1% in likelihood. This gap increases to about 4% after applying classifiers to the inferred ϑ topic distributions, see Table 6.4. The aggregated and limit Gibbs boostings result in negligible deterioration in the likelihood, though they give 5 times speedup.

The observation that much fewer iterations are enough for Gibbs sampling, combined with our Gibbs boosting methods bids fair that LDA may become a computationally highly efficient latent topic model in the future.

6.5.3 Comparison with the baseline

We run supervised web document classification as described in Subsection 6.4.2. The results can be seen in Tables 6.3-6.5, the evaluation metric is *AUC*, averaged over the 11 big categories. Experiments on the big corpus could be carried out only for linked LDA with plain Gibbs sampler, and LDA with various samplers, as it took too much space for the linked LDA model with the Gibbs sampling boostings and too much time for the baseline methods to terminate, see Table 6.3.

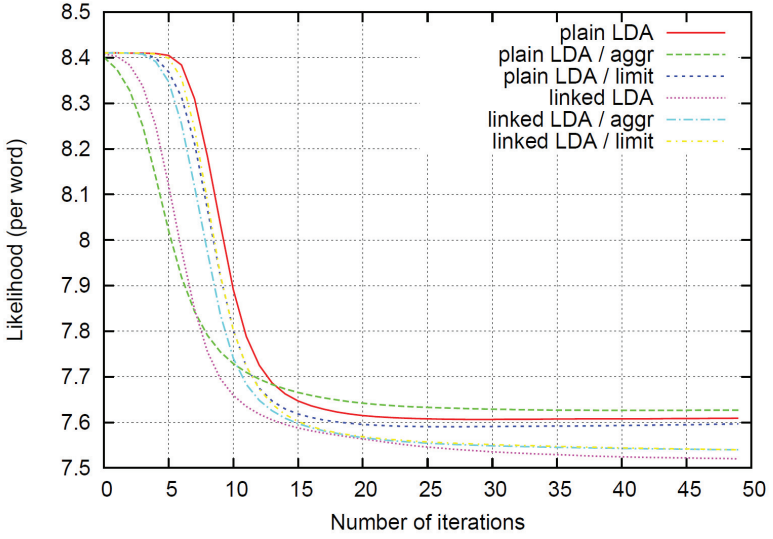

Figure 6.3: Convergence of the likelihood for various models and samplers.

The tables clearly indicate that applying aggregated, limit and sparse Gibbs sampling with sparsity ℓ at most 10, has only a minor negative effect of about 2% on the classification accuracy, albeit they give significant speedup by Table 6.1. Linked LDA slightly outperforms the LDA based categorization for all classifiers, by about 4%. This gap is biggest for BayesNet. The results show an interesting under-performance of SVM, which gives reasonable result only with $tf \times idf$ features.

Table 6.5 indicates that the χ link weights delivered by the linked LDA model captures influence very well, as it improves 2% over $tf \times idf$, 4% over LDA and 3% over linked LDA with the cocitation graph, and 3% over link-PLSA-LDA with its own χ weights. This clearly indicates that the χ link weights provided by the linked LDA model are good approximations of the topical similarity along links. Reversing the graph influences behaves in a quite unpredictable way, though rev-cocit is somewhat better than cocit, furthermore, reversion worsens the *AUC* measure if the weights come from linked or link-PLSA-LDA χ values.

Every run (LDA model build and classification with and without graph stacking) is repeated 10 times to get variance of the *AUC* measure. This was at most 0.015 throughout, so we decided not to quote them individually.

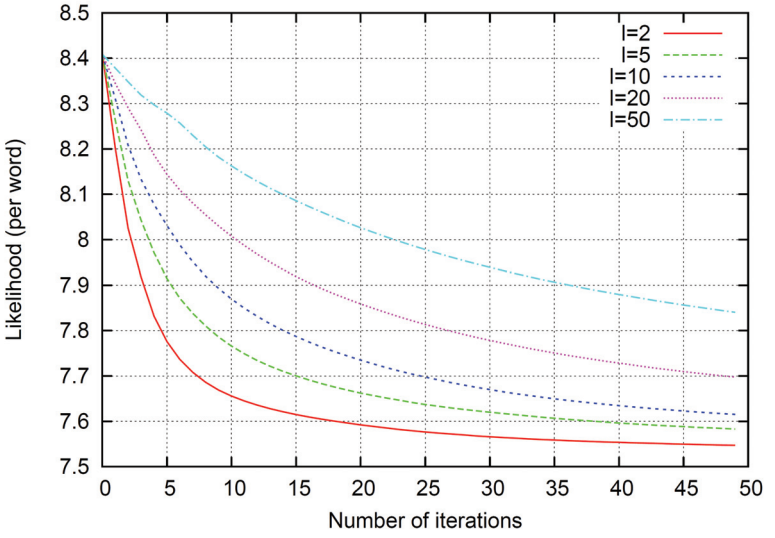

Figure 6.4: Convergence of the likelihood for the sparse sampler for LDA with various sparsity parameters.

6.6 Conclusion and future work

In this chapter we introduced the linked LDA model, which integrates the flow of influence along links into LDA in such a way that each document can be citing and cited at the same time. By our strategies to boost Gibbs sampling, we were able to apply our model to supervised web document classification as a feature generation and dimensionality reduction method. In our experiments linked LDA outperformed LDA and other link based LDA models by about 4% in *AUC*. One of our Gibbs sampler heuristics produced 10-fold speedup with negligible deterioration in convergence, likelihood and classification accuracy. Over our data set of Web hosts, these boostings outperform the fast Gibbs sampler of [91] in speed to a great extent. We also note that our samplers use ideas orthogonal to the fast collapsed Gibbs sampler of [91] and the paralleled sampling of [82], so these methods can be used in combination. It would be interesting to explore other domains than LDA where our Gibbs sampling strategies can be applied. Limit Gibbs sampling makes it possible to have arbitrary non-negative real numbers as word counts in a document, instead of the usual tf counts. To this end, we plan to measure whether accuracy of LDA is improved if the tf counts are replaced with the pivoted $tf \times idf$

model / sampler	BayesNet	SVM	C4.5
LDA	0.821	0.610	0.762
LDA / aggr.	0.820	0.584	0.756
LDA / limit	0.810	0.595	0.739
linked LDA	**0.854**	**0.623**	**0.765**

Table 6.3: Big corpus. Classification accuracy measured in *AUC* for LDA and linked LDA under various Gibbs sampling heuristics.

model / sampler	BayesNet	SVM	C4.5
LDA	0.817	0.605	0.767
LDA / aggr.	0.813	0.591	0.750
LDA / limit	0.808	0.562	0.720
LDA / sparse ($\ell = 2$)	0.805	0.554	0.719
LDA / sparse ($\ell = 5$)	0.799	0.571	0.713
LDA / sparse ($\ell = 10$)	0.791	0.567	0.711
LDA / sparse ($\ell = 20$)	0.764	0.549	0.689
LDA / sparse ($\ell = 50$)	0.735	0.524	0.670
linked LDA	**0.850**	0.609	**0.777**
linked LDA / aggr.	0.849	0.596	0.771
linked LDA / limit	0.845	0.588	0.761
linked LDA / sparse ($\ell = 2$)	0.840	0.583	0.758
linked LDA / sparse ($\ell = 5$)	0.836	0.579	0.753
linked LDA / sparse ($\ell = 10$)	0.827	0.573	0.751
linked LDA / sparse ($\ell = 20$)	0.799	0.556	0.726
linked LDA / sparse ($\ell = 50$)	0.768	0.530	0.705
link-PLSA-LDA/var. inf. [81]	0.827	0.587	0.754
$tf \times idf$	0.569	**0.720**	0.565

Table 6.4: Small corpus. Classification accuracy measured in *AUC* for LDA and linked LDA under various Gibbs sampling heuristics as well as the baseline methods.

counts of [106].

model + graph	BayesNet	C4.5
LDA + cocit	0.830	0.762
LDA + rev-cocit	0.831	0.770
linked LDA + χ	**0.863**	**0.785**
linked LDA + rev-χ	0.857	0.780
linked LDA + cocit	0.838	0.765
linked LDA + rev-cocit	0.839	0.752
link-PLSA-LDA + own-χ	0.839	0.755
link-PLSA-LDA + own-rev-χ	0.837	0.755
link-PLSA-LDA + cocit	0.835	0.754
link-PLSA-LDA + rev-cocit	0.840	0.763
$tf \times idf$ + cocit	0.597	0.589
$tf \times idf$ + rev-cocit	0.595	0.593
$tf \times idf$ + linked LDA-χ	0.611	0.601
$tf \times idf$ + linked LDA-rev-χ	0.609	0.598
$tf \times idf$ + link-PLSA-LDA-χ	0.604	0.597
$tf \times idf$ + link-PLSA-LDA-rev-χ	0.606	0.596

Table 6.5: Small corpus, classification with graph stacking. The base learner may be one of $tf \times idf$, LDA, linked LDA (both without Gibbs sampling heuristics) and link-PLSA-LDA. Edge weights may arise by cocitation or the inferred χ of the linked LDA and link-PLSA-LDA models. These weights may be obtained over the reversed graph, which is indicated by *rev*.

Chapter 7

LDA in Web Spam Filtering

In this chapter we present our experimental results in Web spam filtering using the previously developed LDA models, see Chapters 5 and 6. We participated in the Web Spam Challenge contest in 2008. The goal of the Web Spam Challenge series is to identify and compare Machine Learning methods for automatically labeling structured data represented as graphs, for example a collection of Web sites. The organizers provided us baseline content and link based features, thus we are given a controlled environment to measure the performance of our models. In both cases, we applied feature combination schemes in order to improve classification performance. In case of MLDA, we used a logistic regression based combination, whilst with linked LDA, we combined the results of the different classifiers by the random forest classifier.

We ranked fourth in the competition using the MLDA model. One year later we tested linked LDA on the same corpus and we compared our results to the winner of 2008. Our random forest based combinations achieved an $AUC = 0.854$, whilst the winner's result was $AUC = 0.848$.

Identifying and preventing spam was cited as one of the top challenges in web search engines by [57]. Singhal estimated that the search engine spam industry had a revenue potential of $4.5 billion in year 2004 if they had been able to completely fool all search engines on all commercially viable queries [107]. Due to the large and ever increasing financial gains resulting from high search engine ratings, it is no wonder that a significant amount of human and machine resources are devoted to artificially inflating the rankings of certain Web pages. As all major search engines incorporate anchor text and link analysis algorithms into their ranking schemes, Web spam appears in sophisticated forms that manipulate content as well as linkage [49].

The most widely used definitions of Web spam are as follows. In [49] a content is called spam if it is completely irrelevant for the given search terms.

In another definition [87] search engine spam consists of features that maintainers would not add to their sites if search engines did not exist. Link spam however could act even without the presence of a search engine by misleading users to visit certain pages, in particular for the purpose of misusing affiliate programs.

7.1 Related results

Spam hunters use a variety of content based features [18, 40, 41, 83] to detect web spam, a recent measurement of their combination appears in [20]. Perhaps the strongest SVM based content classification is described in [1]. An efficient method for combining several classifiers is the use of logistic regression, as shown by Lynam and Cormack [71]. In this work we apply a modification of this method for combination: a random forest over the log-odds of the classifiers.

Closest to our methods are the content based email spam detection methods applied to Web spam presented at the Web Spam Challenge 2007 [27]. They use the method of [18] that compresses spam and non-spam separately; features are defined based on how well the document in question compresses with spam and non-spam, respectively. Our method is similar in the sense that we also build separate spam and non-spam content models.

7.2 MLDA as a spam filter

We run the classical LDA separately for both spam and non-spam classes of the training set, then we take the union of the resulting topic collections and make inference w.r.t. this aggregated collection of topics for every unseen document $\tilde{m}$. The total probability of spam topics in the topic distribution of an unseen document gives an *LDA prediction* of being spam or honest.

7.2.1 Experiments

We test the MLDA method in combination with the Web Spam Challenge 2008 public features[1], SVM over pivoted $tf \times idf$ [106], and the connectivity sonar features (analysis of the breadth-first and directory levels within a host together with the internal and external linkage) of [3]. Using logistic regression to aggregate these classifiers, the MLDA method yields an improvement of 11% in *F* and 1.5% in *AUC*. For classification, we used the machine learning toolkit Weka [118]. For a detailed explanation, see Section 5.2.

The data set we used is the UK2007-WEBSPAM corpus. We kept only the labeled sites with 203 labeled as spam and 3589 as non-spam. We aggregated the words and meta keywords appearing in all pages of the sites to form one document per site in a "bag of words" model (only multiplicity and no order information used). We kept only alphanumeric characters and the hyphen but removed all words containing a hyphen not between two alphabetical words.

We used the TreeTagger[2] for stemming and we deleted all stop words enumerated in the

[1] Downloaded from `http://www.yr-bcn.es/webspam/datasets/uk2007/features/` in March 31, 2008.

[2] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`

pair of topic numbers	F	*AUC*
5/50	0.451	0.855
10/50	0.458	0.861
20/50	0.448	0.873
5/10	0.458	0.868
20/20	0.414	0.868

Table 7.1: *F* and *AUC* values of the five best performing LDA predictions.

Onix list[3]. After this procedure the most frequent 22,000 words formed the vocabulary.

We used our own C++ code to run LDA and to run the multi-corpus inference for unseen documents in MLDA. We applied 10-fold cross validation. Two LDA's were run on the training spam and non-spam corpora, and then multi-corpus inference was made to the test documents by Gibbs sampling.

The Dirichlet parameter β was chosen to be constant 0.1 throughout, while $\alpha^{(s)} = 50/k^{(s)}$, $\alpha^{(n)} = 50/k^{(n)}$, and during multi-corpus inference α was constant $50/(k^{(s)} + k^{(n)})$.

We stopped Gibbs sampling after 100 steps for inference on the training data, and after 50 steps for the multi-corpus inference for an unseen document. In a later experiment, we found that only 50-100 iteration is enough to get the same evaluation measures within error bound, for further details see Subsection 6.5.2.

The number of topics were chosen to be $k^{(s)} = 2, 5, 10, 20$ and $k^{(n)} = 10, 20, 50$. Consequently, we performed altogether 12 runs, and thus obtained 12 one-dimensional LDA predictions as features. By observing the *F* curves as shown in Figure 7.1 we selected the five best performing parameter choices. *F* and *AUC* values are shown in Table 7.1. The best result corresponds to the choice $k^{(s)} = 10$ and $k^{(n)} = 50$ with an *F* of 0.46.

Figure 7.1 indicates that the multi-corpus method is robust to the parameter of topic numbers, as the performance does not really change by changing the topic numbers. As one can expect, the maximum of such an *F* curve is approximately $k^{(s)}/k^{(n)}$.

We combined the single best performing $k^{(s)} = 10$, $k^{(n)} = 50$ LDA prediction with an SVM classifier over the $tf \times idf$ features, a C4.5 classifier over the public and the sonar features with logistic regression. All methods were performed by the machine learning toolkit Weka [118]. The *F* and *AUC* values are shown in Table 7.2. LDA improved a relative 11% over the *F* and 1.5% over the *AUC* of the remaining combined features.

We also performed single-feature classification for the 5 best performing LDA predictions over the UK2006-WEBSPAM corpus. Here we have 2125 sites labeled as spam, and 8082 labeled as non-spam. The parameters and the setup was the same as above. The results can be seen at Table 7.3.

[3] `http://www.lextek.com/manuals/onix/stopwords1.html`

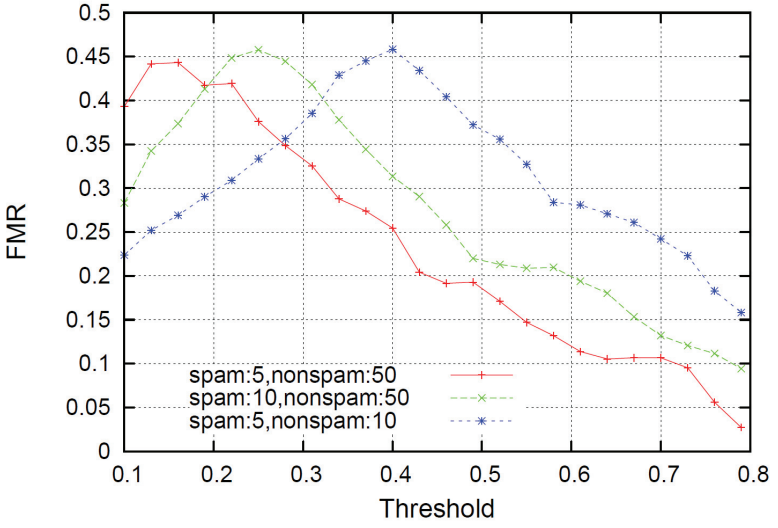

Figure 7.1: F curves of the five best LDA features with varying threshold (horizontal axis).

7.3 Linked LDA as a spam filter

We demonstrate the applicability of linked LDA for Web spam filtering. The inferred topic distributions of documents are used as features to classify the documents as spam or not. To assess the prediction power of these features, we test the linked LDA method in combination with the Web Spam Challenge 2008 public features and SVM over $tf \times idf$. Using a log-odds based random forest to aggregate these classifiers, the inclusion of linked LDA into the public and $tf \times idf$ features yield an improvement of 3% in *AUC*. For a detailed explanation, see Section 6.4.

7.3.1 Experiments

We make experiments on the WEBSPAM-UK2007 corpus[4]. We treat this corpus on site level, that is, we aggregate the words appearing in all html pages of the sites to form one document per site in a "bag of words" model. We keep only alphanumeric characters and the hyphen but

[4] http://barcelona.research.yahoo.net/webspam/datasets/uk2007/

feature set	F	*AUC*
text (SVM)	0.554	0.864
public & text & sonar (log)	0.601	0.954
public & text & sonar & lda (log)	0.667	0.969

Table 7.2: *F* and *AUC* values.

pair of topic numbers	*F*	*AUC*
5–50	0.704	0.881
10–50	0.735	0.902
20–50	0.746	0.919
5–10	0.723	0.906
20–20	0.744	0.925

Table 7.3: *F* and *AUC* values for the UK2006-WEBSPAM corpus.

remove all words containing a hyphen not between two alphabetical words. After stemming by TreeTagger and removing stop words by the Onix list, the most frequent 100k words form the vocabulary.

We take the 6k sites having a Web Spam Challenge 2008 spam or non-spam label, and the sites linked by one of these, to form a corpus of 45k sites, on which the experiments are made. We weight directed links between hosts by their multiplicity, and for every site we keep only at most 10 outlinks with largest weight. After linked LDA inference is run, we have the ϑ topic distribution vectors as features to the classification.

In our experiments, we perform two-class classification for the spamicity. We use the linear kernel SVM, the C4.5 decision tree and the BayesNet implementation of the machine learning toolkit Weka [118].

As the simplest baseline, we use the public features with C4.5 decision tree and the $tf \times idf$ vectors with SVM. For $tf \times idf$, only terms appearing in at least half of the sites are kept. Another baseline is to use the LDA delivered ϑ topic distributions as features with the classifier BayesNet. These baselines and the linked LDA based BayesNet classifier learn on the Web Spam Challenge 2008 train corpus (3900 sites) and are evaluated on the Web Spam Challenge 2008 test corpus (2027 sites), to be able to compare our measurements with the 2008 Challenge results.

Our combination of the classifiers is inspired by the log-odds averaging by Lynam and Cormack [71]. We first make a 10-fold cross validation on the Web Spam Challenge 2008 train corpus to obtain for every site a score for every classifier. Then for every classifier we calculate the log-odds as a feature, which is the logarithm of the fraction of the number of spams with lower score over the number of non-spam sites with higher score. Finally, we learn a random forest over this (quite small) feature set, and evaluate it on the Web Spam Challenge 2008 test

corpus.

For LDA inference the following parameter settings are used. The number of topics is chosen to be $k = 30$ and $k = 90$. The Dirichlet parameter vector components $\vec{\beta}_j$ is constant $200/|V|$, and $\vec{\alpha}_j$ is also a constant with value $50/k$. For a document d, the smoothing parameter vector γ_d is chosen in the same way as in 6.4.3.

We tried three values $p = 1, 4, 10$ in order to find the right smoothing constant for the links.

Gibbs sampling is stopped after 50 iterations, as several measurements indicate that both the *AUC* value on classifying over the topic distributions and the likelihood stabilizes after 50 iterations, see details in Subsection 6.5.2.

The results of the classification can be seen in Tables 7.4-7.6, the evaluation metric is *AUC*. For linked LDA only the parameter choice $p = 4, k = 30$ was included in the combination, as it gave the best result.

	$p = 1$	$p = 4$	$p = 10$
$k = 30$	0.768	**0.784**	0.783
$k = 90$	0.764	0.777	0.773

Table 7.4: Classification accuracy measured in *AUC* for linked LDA with various parameters, classified by BayesNet.

features	*AUC*
LDA with BayesNet	0.766
$tf \times idf$ with SVM	0.795
public (link) with C4.5	0.724
public (content) with C4.5	0.782

Table 7.5: Classification accuracy measured in *AUC* for the baseline methods.

The tables indicate that linked LDA slightly outperforms LDA by about 3% using BayesNet, showing the predicting power of the links in the corpus. Most notably, linked LDA achieved 8% improvement over the public link features with C4.5, and it is at par with the public content features. The addition of linked LDA to the log-odds based combination of the public and $tf \times idf$based classifiers results in a 3% improvement in *AUC*.

As every evaluation was performed on the Web Spam Challenge 2008 test corpus, we can compare these measurements to the 2008 challenge results[5]. The present methods improve a lot over the 0.796 *AUC* achieved by our research group using the MLDA model. The winner, Geng et al., managed to have an *AUC* of 0.848, and even this value is beaten by our public & $tf \times idf$& linked LDA combination, with *AUC* 0.854.

[5] `http://webspam.lip6.fr/wiki/pmwiki.php?n=Main.PhaseIIIResults`

features	*AUC*
$tf \times idf$ & LDA	0.827
$tf \times idf$ & linked LDA	0.831
public & LDA	0.820
public & linked LDA	0.829
public & $tf \times idf$	0.827
public & $tf \times idf$ & LDA	0.845
public & $tf \times idf$ & linked LDA	**0.854**
public & $tf \times idf$ & LDA& linked LDA	**0.854**

Table 7.6: Classification accuracy measured in *AUC* by combining the classifications of Tables 7.4 and 7.5 with a log-odds based random forest. For linked LDA the parameters are chosen to be $p = 4, k = 30$.

7.4 Conclusion and future work

We successfully applied our MLDA and linked LDA models for Web spam filtering. We believe that similar to the success of email spam filtering methods [27], semantic analysis to spam filtering is a promising direction. In future work we plan to implement the email content filtering methods of [27] and test its combination with LDA. As another experiment we are currently measuring the quality of the inferred linked LDA edge weights χ, by using it in a stacked graphical classification procedure, for both the link graph and the cocitation graph.

Chapter 8

References

[1] Jacob Abernethy, Olivier Chapelle, and Carlos Castillo. WITCH: A New Approach to Web Spam Detection. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2008.

[2] E. Alpaydin. *Introduction to machine learning*. MIT press, 2004.

[3] Einat Amitay, David Carmel, Adam Darlow, Ronny Lempel, and Aya Soffer. The Connectivity Sonar: Detecting site functionality by structural patterns. In *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia (HT)*, pages 38–47, Nottingham, United Kingdom, 2003.

[4] N.O. Andrews and E.A. Fox. Recent Developments in Document Clustering. 2007.

[5] C. Andrieu, N. De Freitas, A. Doucet, and M.I. Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50:5–43, 2003.

[6] R. Angelova and G. Weikum. Graph-based text classification: learn from your neighbors. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 485–492, 2006.

[7] Ricardo A. Baeza-Yates. Introduction to data structures and algorithms related to information retrieval. In *Information Retrieval: Data Structures & Algorithms*, pages 13–27. 1992.

[8] András A. Benczúr, Károly Csalogány, and Tamás Sarlós. Link-based similarity search to fight web spam. In *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), held in conjunction with SIGIR2006*, 2006.

[9] Berry, Dumais, and O'Brien. Using linear algebra for intelligent information retrieval. *SIREV: SIAM Review*, 37, 1995.

[10] M.W. Berry. SVDPACK: A Fortran-77 Software Library for the Sparse Singular Value Decomposition. 1992.

[11] I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. *SIAM International Conference on Data Mining*, 2006.

[12] István Bíró, Jácint Szabó, and András A. Benczúr. Latent Dirichlet Allocation in Web Spam Filtering. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2008.

[13] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(5):993–1022, 2003.

[14] D.M. Blei, T.L. Griffiths, M.I. Jordan, and J.B. Tenenbaum. Hierarchical topic models and the nested Chinese restaurant process. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, page 17. Bradford Book, 2004.

[15] Rodrigo A. Botafogo and Ben Shneiderman. Identifying aggregates in hypertext structures. In *Proceedings of the third annual ACM conference on Hypertext*, pages 63–74. ACM Press, 1991.

[16] R. R. Bouckaert. Bayesian network classifiers in weka. Technical Report 14/2004, Comp. Sci. Dept.. U. of Waikato., September 2004.

[17] A. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms, 1997. hardcopy.

[18] A. Bratko, B. Filipič, G.V. Cormack, T.R. Lynam, and B. Zupan. Spam Filtering Using Statistical Data Compression Models. *The Journal of Machine Learning Research*, 7: 2673–2698, 2006.

[19] P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, and R.L. Mercer. Word-sense disambiguation using statistical methods. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pages 264–270. Association for Computational Linguistics Morristown, NJ, USA, 1991.

[20] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: web spam detection using the web topology. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 423–430, 2007.

[21] Carlos Castillo, Kumar Chellapilla, and Ludovic Denoyer. Web spam challenge 2008. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2008.

[22] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. *ACM SIGMOD Record*, 27(2):307–318, 1998.

[23] Soumen Chakrabarti, Mukul M. Joshi, Kunal Punera, and David M. Pennock. The structure of broad topics on the Web. In *Proceedings of the eleventh international conference on World Wide Web*, pages 251–262. ACM Press, 2002.

[24] Jie Cheng, Russell Greiner, Jonathan Kelly, David Bell, and Weiru Liu. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137(1–2):43–90, 2002.

[25] D. Cohn and T. Hofmann. The Missing Link-A Probabilistic Model of Document Content and Hypertext Connectivity. *Advances in Neural Information Processing Systems*, pages 430–436, 2001.

[26] G. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. Technical Report Stanford KSL 9, June 1991.

[27] G.V. Cormack. Content-based Web Spam Detection. In *Proceedings of the 3rd international workshop on adversarial information retrieval, AIRWeb 2007*, Banff, Alberta, Canada, 2007.

[28] Károly Csalogány, A.A. Benczúr, D. Siklósi, and L. Lukács. Semi-Supervised Learning: A Comparative Study for Web Spam and Telephone User Churn. In *Graph Labeling Workshop in conjunction with ECML/PKDD 2007*, 2007.

[29] B. de Finetti and B. de Finetti. Theory of probability, Volume I. *Bull. Amer. Math. Soc. 83 (1977), 94-97. DOI: 10.1090/S0002-9904-1977-14188-8 PII: S*, 2(9904):14188–8, 1977.

[30] Jeffrey Dean and Monika R. Henzinger. Finding related pages in the World Wide Web. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1467–1479, 1999.

[31] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[32] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*, B-39:1–38, 1977.

[33] L. Dietz, S. Bickel, and T. Scheffer. Unsupervised prediction of citation influences. In *Proceedings of the 24th international conference on Machine learning*, pages 233–240. ACM Press New York, NY, USA, 2007.

[34] Stephen Dill, S. Ravi Kumar, Kevin S. McCurley, Sridhar Rajagopalan, D. Sivakumar, and Andrew Tomkins. Self-similarity in the Web. In *The VLDB journal*, pages 69–78, 2001.

[35] Dumais, Platt, Heckerman, and Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM: ACM CIKM International Conference on Information and Knowledge Management*. ACM, SIGIR, and SIGMIS, 1998.

[36] S. Dumais and H. Chen. Hierarchical classification of Web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM New York, NY, USA, 2000.

[37] E. Erosheva, S. Fienberg, and J. Lafferty. Mixed-membership models of scientific publications, 2004.

[38] L. Fei-Fei and P. Perona. A Bayesian hierarchical model for learning natural scene categories. *Proc. CVPR*, 5, 2005.

[39] R. Feldman and I. Dagan. KDT-knowledge discovery in texts. In *Proceedings of the First Annual Conference on Knowledge Discovery and Data Mining (KDD)*, 1995.

[40] Denis Fetterly, Mark Manasse, and Marc Najork. Spam, damn spam, and statistics – Using statistical analysis to locate spam web pages. In *Proceedings of the 7th International Workshop on the Web and Databases (WebDB)*, pages 1–6, Paris, France, 2004.

[41] Denis Fetterly, Mark Manasse, and Marc Najork. Detecting phrase-level duplication on the world wide web. In *Proceedings of the 28th ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, Salvador, Brazil, 2005.

[42] G. Forman, I. Guyon, and A. Elisseeff. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*, 3(7-8): 1289–1305, 2003.

[43] N. Fuhr. *A probabilistic model of dictionary based automatic indexing*. Techn. Hochsch. Darmstadt, Fachbereich Informatik, 1985.

[44] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[45] David Gibson, Jon M. Kleinberg, and Prabhakar Raghavan. Inferring Web communities from link topology. In *UK Conference on Hypertext*, pages 225–234, 1998.

[46] J. Goodman. A bit of progress in language modeling. Technical report, Microsoft Research, 2001.

[47] T.L. Griffiths. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235, 2004.

[48] Steve Gunn. Support vector machines for classification and regression. Technical report, April 07 1998.

[49] Zoltán Gyöngyi and Hector Garcia-Molina. Web spam taxonomy. In *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, Chiba, Japan, 2005.

[50] Zoltan Gyongyi, Hector Garcia-Molina, and Jan Pedersen. Web content categorization using link information. Technical report, Stanford University, 2007.

[51] H. Han, C.L. Giles, E. Manavoglu, H. Zha, Z. Zhang, and E.A. Fox. Automatic document metadata extraction using support vector machines. In *Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*, pages 37–48. IEEE Computer Society Washington, DC, USA, 2003.

[52] J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.

[53] Hanley and McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, April 1982.

[54] Y. Hara and Y. Kasahara. A set-to-set linking strategy for hypertext systems. In *Proceedings of the conference on Office information systems*, pages 131–135. ACM Press, 1990.

[55] David Heckerman. Tutorial on learning in bayesian networks. Technical Report MSR-TR-95-06, Microsoft, 1995.

[56] G. Heinrich. Parameter estimation for text analysis. Technical report, Technical Report, 2004.

[57] Monika R. Henzinger, Rajeev Motwani, and Craig Silverstein. Challenges in web search engines. *SIGIR Forum*, 36(2):11–22, 2002.

[58] T. Hofmann. Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Machine Learning*, 42(1):177–196, 2001.

[59] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, New York, NY, USA, 1999. ACM.

[60] Andreas Hotho, Andreas Nürnberger, and Gerhard Paass. A brief survey of text mining. *LDV Forum*, 20(1):19–62, 2005.

[61] David Hull. Improving text retrieval for the routing problem using latent semantic indexing. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, pages 282–291, London, UK, July 1994. Springer Verlag.

[62] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. Technical Report LS VIII-Report, Universität Dortmund, Dortmund, Germany, 1997.

[63] Zhanzhen Kou. *Stacked graphical learning*. PhD thesis, School of Computer Science, Carnegie Mellon University, December 2007.

[64] Zhenzhen Kou and William W Cohen. Stacked graphical models for efficient inference in markov random fields. In *SDM 07*, 2007.

[65] D. Kuropka. Modelle zur Repräsentation natürlichsprachlicher Dokumente–Information-Filtering und-Retrieval mit relationalen Datenbanken. *Advances in Information Systems and Management Science*, 10, 2004.

[66] Edda Leopold and Jörg Kindermann. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, 46(1/3):423, 2002.

[67] J. Li and M. Sun. Scalable Term Selection for Text Categorization. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 774–782, 2007.

[68] S.H. Lin, M.C. Chen, J.M. Ho, and Y.M. Huang. ACIRD: intelligent Internet document organization and retrieval. *IEEE Transactions on knowledge and data engineering*, 14 (3):599–614, 2002.

[69] J.S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2001.

[70] Nicolas Loeff. Latent dirichlet allocation. Presentation.

[71] T.R. Lynam, G.V. Cormack, and D.R. Cheriton. On-line spam filter fusion. *Proc. of the 29th international ACM SIGIR conference on Research and development in information retrieval*, pages 123–130, 2006.

[72] D.J.C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[73] I. Mani and P. Code. Automatic summarization. *Computational Linguistics*, 28(2), 2001.

[74] C.D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[75] C.D. Manning, P. Raghavan, and H. Schtze. *Introduction to information retrieval*. Cambridge University Press New York, NY, USA, 2008.

[76] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, England, 2000.

[77] Massimo Marchiori. The quest for correct information on the Web: hyper search engines. In *Selected papers from the sixth international conference on World Wide Web*, pages 1225–1235. Elsevier Science Publishers Ltd., 1997.

[78] M. E. Maron. Automatic indexing: An experiment inquiry. *Journal of the ACM*, 8(3): 404–417, July 1961.

[79] M.Steyvers and T. Griffiths. Probabilistic topic models. In S. Dennis T. Landauer, D.S. McNamara and W. Kintsch, editors, *Handbook of Latent Semantic Analysis*. Mahwah, NJ: Erlbaum, 2007.

[80] R. Nallapati and W. Cohen. Link-plsa-lda: A new unsupervised model for topics and influence in blogs. In *International Conference for Weblogs and Social Media*, 2008.

[81] R. Nallapati, A. Ahmed, E. Xing, and W.W. Cohen. Joint Latent Topic Models for Text and Citations. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press New York, NY, USA, 2008.

[82] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed Inference for Latent Dirichlet Allocation. *Advances in Neural Information Processing Systems*, 20, 2007.

[83] Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. Detecting spam web pages through content analysis. In *Proceedings of the 15th International World Wide Web Conference (WWW)*, pages 83–92, Edinburgh, Scotland, 2006.

[84] D.W. Oard. The state of the art in text filtering. *User Modeling and User-Adapted Interaction*, 7(3):141–178, 1997.

[85] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford University, 1998.

[86] B. Pang and L. Lee. *Opinion mining and sentiment analysis*. Now Publishers, 2008.

[87] Alan Perkins. White paper: The classification of search engine spam, September 2001. Online at http://www.silverdisc.co.uk/articles/spam-classification/ (visited June 27th, 2005).

[88] James Pitkow and Peter Pirolli. Life, death, and lawfulness on the electronic frontier. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 383–390. ACM Press, 1997.

[89] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

[90] A. Popescul, L. H. Ungar, D. M. Pennock, and S. Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the 17th International Conference on Uncertainty in Artificial Intelligence*, pages 437–444. Morgan Kaufmann, San Francisco, CA, 2001.

[91] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast Collapsed Gibbs Sampling for Latent Dirichlet Allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press New York, NY, USA, 2008.

[92] M. Porter. Snowball stemming algorithm.

[93] M. F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 3 1980.

[94] J.K. Pritchard, M. Stephens, and P. Donnelly. Inference of Population Structure Using Multilocus Genotype Data. *Genetics*, 155(2):945–959, 2000.

[95] Xiaoguang Qi and Brian D. Davison. Knowing a web page by the company it keeps. In *Proceedings of the 15th Conference on Information and Knowledge Management (CIKM)*, 2006.

[96] J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[97] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA., 1993.

[98] K. Rose, E. Gurewitz, and GC Fox. A Deterministic Annealing Approach To Constrained Clustering. *Information Theory, 1991 (papers in summary form only received), Proceedings. 1991 IEEE International Symposium on (Cat. No. 91CH3003-1)*, pages 373–373, 1991.

[99] S. Russell and P.N.A. Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 1995.

[100] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[101] G. Salton, A. Wong, and A. C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18:229–237, 1975.

[102] M. Sanderson. Word sense disambiguation and information retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 142–151. Springer-Verlag New York, Inc. New York, NY, USA, 1994.

[103] S. Sarawagi. Information extraction. *Foundations and Trends® in Databases*, 1(3): 261–377, 2007.

[104] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, volume 12. Manchester, UK, 1994.

[105] Fabrizio Sebastiani. Machine learning in automated text categorization. *CSURV: Computing Surveys*, 34, 2002.

[106] A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document length normalization. *Information Processing and Management*, 32(5):619–633, 1996.

[107] Amit Singhal. Challenges in running a commercial search engine. In *IBM Search and Collaboration Seminar 2004*. IBM Haifa Labs, 2004.

[108] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–29, New York, NY, USA, 1996. ACM.

[109] J. Sinkkonen, J. Aukia, and S. Kaski. Component models for large networks. *Arxiv preprint arXiv:0803.1628*, 2008.

[110] J. Sivic, BC Russell, AA Efros, A. Zisserman, and WT Freeman. Discovering Objects and their Localization in Images. *Computer Vision, ICCV 2005. Tenth IEEE International Conference on*, 1, 2005.

[111] Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[112] Y.W. Teh, D. Newman, and M. Welling. A Collapsed Variational Bayesian Inference Algorithm for Latent Dirichlet Allocation. *Advances in Neural Information Processing Systems*, 19:1353, 2007.

[113] D. Tikk, editor. *Szövegbányászat*. TypoTEX, Budapest, 2007. 294 pages.

[114] K. Tzeras and S. Hartmann. Automatic indexing based on Bayesian inference networks.

[115] D.D. Walker and E.K. Ringger. Model-based document clustering with a collapsed gibbs sampler. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press New York, NY, USA, 2008.

[116] Ron Weiss, Bienvenido Vélez, and Mark A. Sheldon. HyPursuit: a hierarchical network search engine that exploits content-link hypertext clustering. In *Proceedings of the the seventh ACM conference on Hypertext*, pages 180–193. ACM Press, 1996.

[117] J. H. Wilkinson and C. Reinsch. *Handbook for Automatic Computation. Vol. II Linear Algebra*. Springer-Verlag, New York, 1971.

[118] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005.

[119] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco, 2005.

[120] D. Xing and M. Girolami. Employing Latent Dirichlet Allocation for fraud detection in telecommunications. *Pattern Recognition Letters*, 28(13):1727–1734, 2007.

[121] F. Yamout, R. Demachkieh, G. Hamdan, and R. Sabra. Further enhancement to Porter algorithm. In *Proceedings of the KI2004 Workshop on Machine Learning and Interaction for Text-based Information Retrieval, Germany*, pages 7–24, 2004.

[122] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, 1997.

[123] H. Zhang, C.L. Giles, H.C. Foley, and J. Yen. Probabilistic Community Discovery Using Hierarchical Latent Gaussian Mixture Model. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 663. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[124] H. Zhang, B. Qiu, C.L. Giles, H.C. Foley, and J. Yen. An LDA-based Community Structure Discovery Approach for Large-Scale Social Networks. *Intelligence and Security Informatics, 2007 IEEE*, pages 200–207, 2007.

[125] X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. *Advances in Neural Information Processing Systems*, 17:1641–1648, 2005.

[126] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.